RICE UNIVERSITY

A Compressive Phase-locked Loop

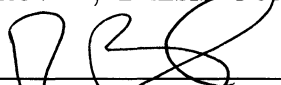
by

Stephen Schnelle

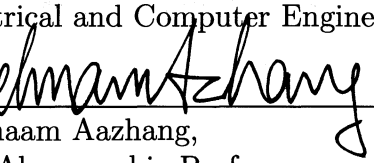
A THESIS SUBMITTED
IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE

Master of Science

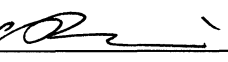
APPROVED, THESIS COMMITTEE:



Richard G. Baraniuk, Chair,
Victor E. Cameron Professor,
Electrical and Computer Engineering



Behnaam Aazhang,
J.S. Abercrombie Professor,
Electrical and Computer Engineering



Wotao Yin,
Assistant Professor,
Computational and Applied Mathematics

HOUSTON, TEXAS

APRIL 2011

Abstract

We develop a new method for tracking narrowband signals acquired through compressive sensing, called the compressive sensing phase-locked loop (CS-PLL). The CS-PLL enables one to track oscillating signals in very large bandwidths using a small number of measurements. Not only does the CS-PLL potentially operate below the Nyquist rate, it can extract phase and frequency information without the computational complexity normally associated with compressive sensing signal reconstruction.

The CS-PLL has a wide variety of applications, including but not limited to communications, phase tracking, robust control, sensing, and FM demodulation. In particular we emphasize the advantages of using this system in wideband surveillence systems. Our design modifies classical PLL designs to operate with CS-based sampling systems. Performance results are shown for PLLs operating on both real and complex data. In addition to explaining general performance tradeoffs, implementations using several different CS sampling systems are explored.

Acknowledgements

This thesis was written with the support of numerous people, to each of which I am extremely appreciative. First off, I would like to thank Rich for his inspiration and energy. He offered much advice, both for this paper as well as navigating the challanges of graduate school.

Petros Boufounos and Mark Davenport also provided many insightful conversations and ideas for this work.

JP Slavinksy's contributions also cannot go unmentioned, for all his time spent in technical discussion, organization, motivation, emotional support, and many other areas.

Past and present members of Rice DSP Research Group, including Eva Dyer, Manjari Narayan, Jason Laska, Chinmay Hegde, Marco Duarte, Andrew Waters, Mona Sheikh, and Sriram Nagaraj have also provided much help, stimulating ideas through conversations in group meetings, late night office discussions, and many other times.

Lastly, I thank my friends, my parents and sister for their continuing love, and last but definitely not least, God.

Contents

List of Figures

Chapter 1

Introduction

Recent results in signal processing have demonstrated that appropriately exploiting the structure of a signal can result to significant reduction in the acquisition rate. Compressive sensing (CS) is the most celebrated of those results [1,4,13]. CS demonstrated that if a signal is sparse, it can be acquired without loss of information at a rate closer to its number of nonzeros, rather than the Nyquist rate, using a randomized measurement system. The signal can then be reconstructed by solving a convex optimization problem or using a greedy sparse recovery algorithm.

The advent of CS has spawned a number of signal acquisition hardware architectures that implement the randomized measurement systems recommended by the theory. These include the single-pixel camera and the Coded Aperture Snapshot Spectral Imaging (CASSI) for the sensing of images [15, 44], and the random demodulator [41], the random sampler [26], the random filter [42], and the compressive multiplexer [33] for the acquisition of wideband analog signals. Using these architectures, a variety of signals can be acquired, and reconstructed using the standard CS reconstruction algorithms.

Most sparse reconstruction algorithms are relatively expensive computationally and operate on finite-dimensional windows of data. While this is acceptable in many settings, it creates several problems in streaming applications such as radio receivers and video acquisition. In these scenarios low computational complexity and real-time reconstruction is paramount. Furthermore, the finite-dimensional nature of reconstruction algorithms requires that streaming signals are processed in finite-length blocks, introducing significant input-output delay and blocking artifacts in the block boundaries. Thus, classical CS recovery algorithms are not appropriate for applications requiring frequency and phase tracking, such as demodulation of frequency modulated (FM) signals.

Some approaches exist to address the complexity of standard CS reconstruction algorithms by directly extracting the required information from a resulting low dimensional data set. For example, background subtraction can be used to find differences in the compressive data, equating (within error) to differences in the Nyquist data, useful in anomaly detection such as surveillance and structural health monitoring [8]. Another example is applying matched filtering to CS data, applicable in classification and detection problems [16].

In this paper we introduce a phase-locked loop (PLL) architecture, the CS-PLL, designed extract information directly from compressively sensed streaming signals. The CS-PLL does not require signal reconstruction; it operates in the compressive domain. Thus it has minimal complexity and does not require block-processing of the

incoming signal. The CS-PLL is able to track the frequency and phase of compressively acquired signals for the purposes of control and/or FM decoding. For example, combined with filtering and detection systems, the CS-PLL is a key component of the wideband compressive radio receiver [12], which can track and monitor a wide range of radio frequencies in real time, using inexpensive compressive signal acquisition hardware.

In acquiring and tracking signal frequency an phase from compressively acquired signals, the CS-PLL has two distinct advantages over conventional CS approaches. First, it operates in real-time on streaming signals, compared to the block-based approaches in most CS reconstruction systems. Second, it has very low computational complexity, almost as low as a conventional PLL. Thus, in tracking frequency and phase, the CS-PLL also outperforms streaming CS reconstruction algorithms, such as [3, 30], since these algorithms would require the use of a conventional PLL in the reconstructed signal. [43] explores the application of a Kalman filter to CS data by updating the support set as necessary and running a Kalman filter on the determined support set until it changes and the error signal becomes too large, but does not delve into Kalman filter models for specific applications such as compressively sampled FM data.

The CS-PLL achieves these advantages by operating directly in the compressive domain. In doing so it exploits the property of compressive measurements to preserve the structure of the acquired signals. This property has led to a number of

compressive-domain processing algorithms for detection, estimation and filtering [10]. The CS-PLL exploits this property to implement the architecture of a conventional PLL in the compressive domain, thus exploiting the straightforward and simple computational structure of conventional PLLs.

A subset of the CS-PLL includes PLLs that use random sampling. We note that non-uniform sampling PLLs have been explored in a few other previous works from different angles. Basic theoretical results of additive random sampling (ARS) and quantized ARS (in the context used, quantized ARS refers to discretized sampling times, not that the samples themselves are limited in precision) on digital lock-in amplifiers and were addressed in [31, 34–36], supplemented with a simple FPGA implementation of a phase-locked loop for quantized ARS. Moreover, the basic effects of non-uniform sampling in the context of sampling jitter in PLLs have been analyzed [46]. Recently [37] provides numerical simulations for varying loop filter parameters in a random sampling PLL as well, though the effects of noise and compression ratio on SNR are not detailed; rather a more idealistic approach for a synchronization application is used.

In the next section we establish some notation and provide relevant background on CS and PLLs. Chapter 3 introduces the CS-PLL, while Chapter 4 provides an analysis of it. Chapter 5 addresses the issues with designing a Kalman filter version of the CS-PLL. Chapter 6 addresses practical issues that allow the CS-PLL to be integrated into a larger system. Chapter 7 provides experimental results of the system. Chapters 8

and 9 discuss some variants of the CS-PLL and Chapter 10 concludes the paper.

Chapter 2

Background

2.1 Traditional Phase-Locked Loop

The phase-locked loop (PLL) is a well-established method for tracking a signal's frequency and phase [20]. It is a continuous-time or discrete-time architecture that uses a feedback loop to continuously update an estimate of the frequency and/or phase of an input signal by generating a reference signal, comparing the phase of the input with the reference, and adjusting the reference until the signals match according to some measure, in which case the PLL is referred-to as locked to the input signal. Either the generated reference signal or its phase information can be used as output depending on the application. For example, for FM demodulation, the phase signal is utillized and filtered to reconstruct the underlying modulated signal.

Figure 2.1 shows a typical discrete-time real-valued PLL architecture. The PLL tracks the phase of a discrete-time input signal $x[n]$, by adjusting the phase of a generated reference signal $u[n]$ until the reference signal is approximately orthogonal to the input. The tracking is performed through a feedback loop with three fundamental components: an oscillator, a phase detector, and a loop filter [17]. The phase detector

and loop filter estimate the phase difference between $x[n]$ and $u[n]$ by multiplying the two signals and low-pass-filtering the product to obtain

$$\theta[n] = \sum_k x[k]\, u[k]\, h[n-k], \tag{2.1}$$

where $h[n-k]$ is the impulse response of the low-pass filter. The phase estimate is used by the oscillator to produce the reference signal

$$u[n] = \cos(2\pi f n + \theta[n]), \tag{2.2}$$

which is compared with the input signal. The architecture is similar for complex-values signals [23].

The phase detector output $\widehat{\theta}[n]$ followed by the low-pass filter can be determined as the kernel inner product of the incoming signal with the generated one, with the kernel defined by the low-pass filter. If the signals are orthogonal, then the inner product is zero, and we consider the PLL locked. Otherwise the inner product will be positive or negative depending on which of the two signals leads or lags in the phase term. The magnitude of this inner product provides an estimate of the corresponding lead or lag. The goal of the phase update is to use the output of the inner product to update the phase estimate, if necessary. The phase error is called loop stress; an expected loop stress of zero indicates that the loop is unbiased.

As an example, if the input is

$$x[n] = \sin(2\pi f_0 t + \theta_0(t)),$$

we want our oscillator to produce

$$u[n] = \cos(2\pi f_1 t + \theta_1(t)),$$

where $f_0 \approx f_1$ and $\theta_0 \approx \theta_1$. The phase detector product yields a low frequency component

$$\frac{1}{2}\sin(2\pi(f_0 - f_1)t + (\theta_1(t) - \theta_2(t))),$$

in which we are interested (approximated as $\frac{1}{2}\sin(\theta_1(t) - \theta_2(t))$ using the small angle approximation), and a high frequency component

$$\frac{1}{2}\sin(2\pi(f_0 + f_1)t + (\theta_1(t) + \theta_2(t)))$$

which is removed by the loop filter. If the frequency of the two signals, f_0 and f_1 are the same, the output is 0 if the signals have a phase difference of zero or π. Otherwise, the output dictates how θ_1 should change to approach θ_0. Given a well-designed loop filter, sufficiently low delay in the system, and reasonable SNR, the system should achieve stable.

A simple second order loop enables the PLL to track phase and frequency while

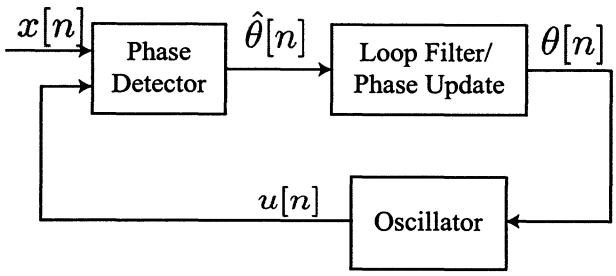


Figure 2.1: Basic discrete time PLL design.

a higher order system is useful for handling Doppler effects. We use a loop filter described by the transfer function

$$H_l(z) = C_2 \frac{(z-1) + \frac{C_1}{C_2}}{(z-1)} \tag{2.3}$$

as in [38],where $C_1 = \omega_n^2$, and $C_2 = 2\zeta\omega_n$, for comparison. (Including the phase update integrator present after the filter yields the response $H(z) = C_2 \frac{(z-1)+\frac{C_1}{C_2}}{(z-1)^2}$ with two poles and hence a second-order PLL). A second order PLL is normally sufficient for locking onto carrier frequencies that are typically not exactly known, whereas a first order PLL can only track phase. We choose to use the simple, low bandwidth sinusoid carrier function instead of other such as square wave modulation that produce additional frequency components.

The PLL lock and the deviation from it is used differently in the variety of applications. For applications such as FM demodulation, when the correct frequency is maintained, the deviation in phase lock encodes the desired message signal. For clock synchronization applications, there should be no deviation from phase or frequency lock.

2.2 Compressive Sensing and the Restricted Isometry Property

In the fundamental CS framework, we acquire a signal $x \in \mathbb{R}^N$ as

$$y = \Phi x, \tag{2.4}$$

with Φ an $M \times N$ matrix representing the sampling system and $y \in \mathbb{R}^M$ the vector of acquired measurements. Throughout most of the paper we assume continuous-valued measurements y. Uniform sampling theory requres M samples equal to the signal dimension N to ensure no information is lost, while CS theory, allows the acquisition of significantly fewer samples, given that the signal x is sparse or compressible in some basis [5–7, 14].

This reduction in measurements can be explained by the properties of Φ, in particular the *restricted isometry property* (RIP) introduced by Candès and Tao [6]. First we define Σ_K to be the set of all K-sparse signals in $\mathbb{R}^N$, i.e.,

$$\Sigma_K = \{x \in \mathbb{R}^N : \|x\|_0 \leq K\},$$

where $\|\cdot\|_0 := |supp(x)|$ simply counts the number of non-zero entries of a vector. We define the RIP of order K of a matrix Φ such that the matrix satisfies this property if the relationship

$$(1-\delta)\|x\|_2^2 \leq \|\Phi x\|_2^2 \leq (1+\delta)\|x\|_2^2 \tag{2.5}$$

holds for some constant $\delta \in (0,1)$ over all $x \in \Sigma_K$, (*i.e.* Φ acts as an approximate isometry on the set of vectors that are K-sparse).

A key consequence of the RIP is that inner products are approximately preserved between any two sparse or compressible signals:

$$\langle \Phi x_1, \Phi x_2 \rangle \approx \langle x_1, x_2 \rangle. \tag{2.6}$$

This property ensures that the signal geometry between the signals of interest is preserved by the sampling process, i.e., (2.6) is an equality up to a small difference, $\pm\eta$, that can be controlled by the number of measurements. In particular, [11] shows that for a matrix Φ satisfying the Restricted Isometry Property of order $\max(\|x_1 + x_2\|_0, \|x_1 - x_2\|_0)$ with isometry constant δ, the following relationship holds:

$$|\langle \Phi x_1, \Phi x_2 \rangle - \langle x_1, x_2 \rangle| \leq \delta \|x_1\|_2 \|x_2\|_2 \tag{2.7}$$

2.3 Compressive Samplers in Practice

CS sampling methods include *random demodulation* [24,27], an architecture based on a wideband pseudorandom modulator and a low-rate sampler, *random sampling* [19], an architecture based on pseudo-random non-uniform time samples, and CMUX [33]. All of these systems can efficiently acquire a large class of compressible signals.

2.3.1 Random demodulator

The architecture of the random demodulator [24, 27] is depicted in Figure 2.2. An analog input $x(t)$ is modulated with a pseudo-random stream of square pulses with amplitude ± 1s, called the *chipping sequence* $p_m(t)$, with transition frequency at or above the Nyquist rate N_aHz of the input signal. Next, integration over a time period $1/M_a$ is performed on the mixed signal, and lastly this result is sampled by a back-end ADC at $M_a\text{Hz} < N_a\text{Hz}$. Practically, data over a period T is processed for each sample with $N = N_a T$ the number of elements in the chipping sequence, and $M = M_a T$ the number of measurements. In hardware it is far easier to build a high-rate modulator/chipping sequence combination than a high-rate ADC, so we benefit from allowing the ADC to work at a low rate. Indeed, CDMA, BPSK, and other communication schemes already use these components.

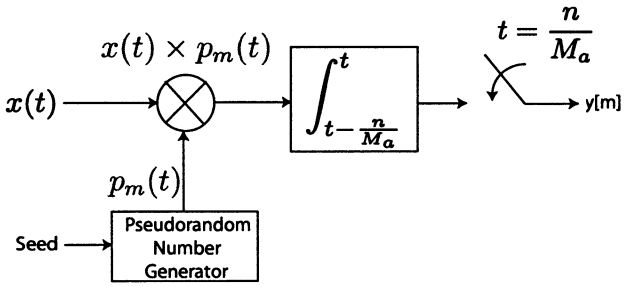


Figure 2.2: Random demodulator block diagram.

This system corresponds to a discretized model of multiplying a sampled Nyquist-rate signal x with a random sequence of ± 1s and summing every N/M product values. We can represent it with a banded matrix Φ operating on x where each row contains N/M pseudo-random ± 1s. For example, with $N = 9$ and $M = 3$, such a Φ

is expressed as

$$\Phi = \begin{bmatrix} 1 & -1 & -1 & & & & & & \\ & & & -1 & -1 & 1 & & & \\ & & & & & & 1 & 1 & -1 \end{bmatrix}.$$

We also say that this system has a window length $L = 3$, as 3 Nyquist rate components are acquired per compressive sample. In the case of a single random demodulator $N/M = L$. $p_m[k]$ represents element (m, k) of this sampling matrix (indexed at (0,0)).

A discrete-time formulation of this model is shown in Fig 2.3. We replace the analog chipping sequence with a sampled chipping sequence, the multipler with a discrete ideal multiplier, and the integrate-and-dump system with an accumulate-and-dump system.

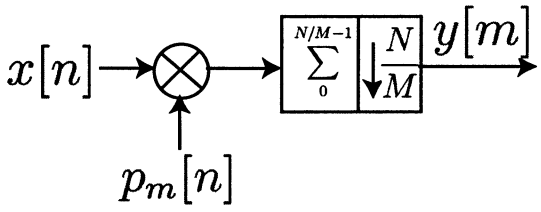


Figure 2.3: Discrete-time formulation of random demodulator.

We can also use several random demodulator together in a system, offseting the periods of the downsampler components, so that the signal information at any time instant is included in multiple compressive samples. We refer to this as interleaved random demodulators and can construct a similar matrix representation as before. For example, a system with two random demodulators each with a compression factor

of 4 can be represented as

$$\Phi = \begin{bmatrix} -1 & 1 & -1 & 1 & & & & \\ & & -1 & -1 & -1 & 1 & & \\ & & & & 1 & 1 & 1 & -1 \end{bmatrix}$$

Here, the total compression factor N/M is 2 (although the total number of number rows N divided by the total number of columns M is not exactly 2, this is because the first and last rows cannot overlap with previous and next samples respectively; asymptotically as the number of measurements is increased we obtain a compression factor of 2). Unlike the single random demodulator case where $N/M = L$, the window length (number of Nyquist rate components acquired per compressive sample) L is now 4. We also note that sample m at the output of compressive sampler using interleaved random demodulators will denote the m^{th} measurment of the overall system, not an individual random demodulator, so $p_m[k]$ again denotes element (m, k) of this sampling matrix (indexed at (0,0)).

Instead of pulses with amplitude determined by a Bernoulli random variable (± 1), we could also use pulses with amplitude determined by a Gaussian random variable (such that the non-zero coefficients in our discrete formulation are Gaussian distributed). While this may not be practical, it can slightly improve system performance.

2.3.2 Random sampling

CS fundamentally requires that the sampling function Φ be incoherent with the signal basis Ψ in which the signal is sparse. Since the Fourier basis is maximally incoherent with the canonical basis, signals with sparse frequency spectra can be reconstructed from non-uniform random time samples [7]. For example, with $N = 9$ and $M = 3$, one such resulting matrix Φ is

$$\Phi = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}.$$

Again $p_m[k]$ denotes element (m, k) of this sampling matrix, with each coefficient either 0 or 1.

Practical implementations may have some periodicity of sample spacing but appear non-uniform within each period, for example when implemented as a bank of parallel low-rate samplers out of phase with each other. These low-rate samples could also use different rates as well. Another implementation is a high-rate Nyquist sampler that does only store and/or transmits some of the samples. However this alternative does not provide cost benefits in the ADC hardware, which is often the bottleneck when working with both analog and digital system components.

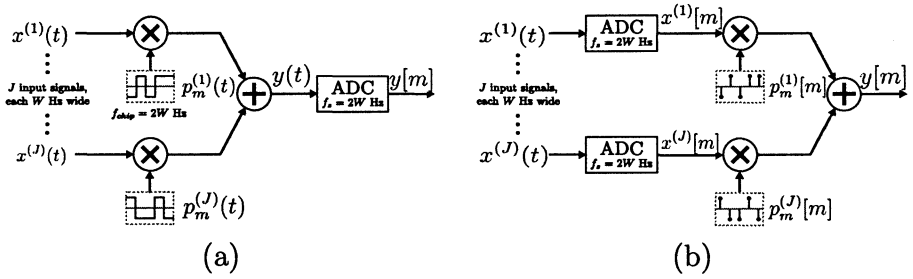


Figure 2.4: (a) Ideal CMUX system diagram. Each of the J channels is spread by a different chipping sequence, then summed and sampled. (b) Ideal CMUX equivalent system.

2.3.3 Compressive Multiplexer (CMUX)

The CMUX combines J non-overlapping signal channels into a single stream of samples [33]. These independent channels have equal bandwidth $W/2$ Hz. Each channel is modulated by an independent pseudo-random ± 1 chipping sequence $p_m^{(j)}(t)$ (where j denotes one of J channels) with chipping frequency W Hz, and as shown in Figure 2.4(a), combined to produce a stream at this Nyquist rate. Only one ADC is necessary for this system despite an analytically equivalent system shown in Figure 2.4(b) using one per channel. The fundamental difference between the CMUX architecture and many other CS sampling systems is that linear combinations are produced from components over multiple channels, rather than from a single signal over time.

A discrete-time formulation of the CMUX architecture consists of a $W \times JW$ matrix Φ formed by concatenating J diagonal $W \times W$ submatrices Φ_j. Similar to our examples above for the random demodulator and random sampler with compression

of 3, we let $J = 3$ and $W = 3$. We could express Φ as

$$\Phi = \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\Phi_1} \underbrace{\begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\Phi_2} \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix}}_{\Phi_3}$$

For each Φ_j, we can write $p_m^{(j)}[m]$ corresponding the m^{th} non-zero coefficient of channel j. As we show in Section 9.1, if we wish to demodulate a signal in band j, we will use the pseudorandom sequence $p_m^{(j)}[m]$ corresponding to that channel j.

The sparsity of the system is determined by the collective sparsity of each channel, and a $JW \times JW$ block diagonal matrix with $W \times W$ DFT bases along the diagonal acts a basis for the overall system. CS reconstruction algorithms could then solve for a K-sparse vector $\alpha \in \mathbb{R}^{JW}$, constraining $y = A\alpha$ and defining A as

$$A = [\Phi_1 \mathcal{F}, \Phi_2 \mathcal{F}, \cdots, \Phi_J \mathcal{F}], \tag{2.8}$$

with $\mathcal{F}$ representing a unitary DFT matrix of size $W \times W$.

2.4 Noise Amplification

It is important to highlight the inherent input noise amplification of CS [40], as the same issues arise in both traditional subsampling systems as well as the CS-PLL. We rely on the original blockwise finite-time formulation of CS, but the fundamental

results apply for streaming wideband signals. We consider that the N-dimensional signal x has been distorted with AWGN, denoted as w, with covariance matrix $\mathrm{K}_w = \mathrm{I}_N$ before sampling. If x is a K-sparse signal in an N-dimensional space with support set S ($|S| = K$), we find that noise is aliased into the support. We make an important distinction between noise added prior and after acquistion. Noise on the compressive measurements is not nearly as detrimental and was the primary case considered in the original CS results. Suppose we acquire the signal with Φ, an $M \times N$ compressive sampling matrix satisfying RIP and normalized with each row of norm $\sqrt{\frac{N}{M}}$. To obtain ideal results, we also assume all the columns are orthogonal to each other.

By constraining Φ in this manner we know that

$$\Phi\Phi^T = \frac{N}{M}\mathrm{I}_M \tag{2.9}$$

and

$$\Phi_S^T\Phi_S = \mathrm{I}_K \tag{2.10}$$

When we take compressive meassurements of the noisy signal, we find that

$$\begin{aligned} \mathbf{y} &= \Phi(x + w) \\ &= \Phi_S x_S + \Phi w \end{aligned} \tag{2.11}$$

This yields a covariance matrix for the noise at this stage as $\mathrm{K}_w = \Phi\Phi^T = \frac{N}{M}\mathrm{I}_M$. Hence the noise is amplified. If we continue with the reconstruction process, to

find the best case reconstruction we assume full knowledge of the support set S. A least squares estimate can then be obtained using the pseudo-inverse of Φ_S, denoted as $\Phi_S^\dagger$. This results in a reconstructed $\hat{x}$ of $x_S + \Phi_S^\dagger \Phi w$. The covariance of the noise after reconstruction is $K_{\hat{w}} = (\Phi_S^\dagger \Phi)(\Phi_S^\dagger \Phi)^T$, which simplifies to $\frac{N}{M} \mathrm{I}_K$, indicating that the resulting noise after reconstruction is similarly amplified by the compression factor. This implies that we will never be able to avoid an output SNR reduction as we increase the compression factor, equivalent to 3dB per compression by additional factor of 2. However, as we reduce our input SNR, support recovery begins to get worse and our performance deviates more from this upper limit.

Figure 2.5 shows an example reconstruction SNR when running the traditional CS reconstruction algorithm CoSAMP. We have a 60 dB input SNR signal where several of the bands would overlap if the signal were subsampled. We find that the locations of the bands does not degrade performance using CS, but as the signal is continually compressed we lose roughly 3dB per factor of 2 compression due to noise. At a certain point, as we take too few measurements, we no longer have sufficient information to perform reconstruction and we veer dramatically from the performance limit. In the example, we see that compression by 64 (2^6) is too much. The "oracle" curve indicates the additional bound on performance; while traditional CS must determine both the support and amplitude of coefficient, the "oracle" bound is the least squares estimation over the known support to determine coefficient amplitudes, deviating much less from the performance limit.

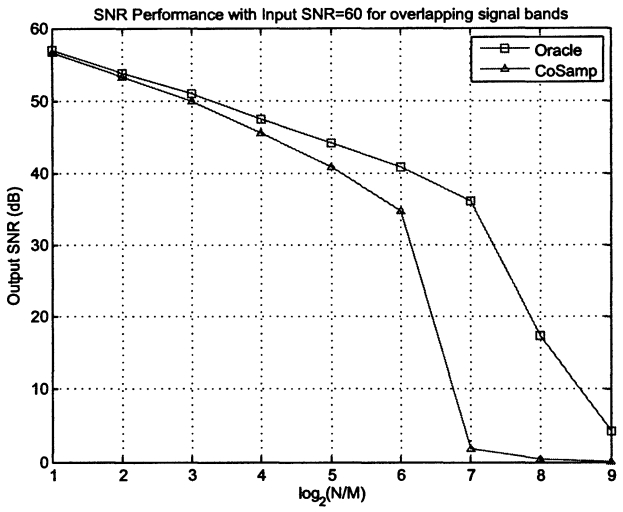


Figure 2.5: SNR of CoSamp reconstruction of sparse signal as we vary compression factor showing noise amplification

Chapter 3

Compressive Sensing PLL (CS-PLL)

Standard PLL designs in the literature operate under the assumption that the input signal is sampled uniformly at or above the Nyquist rate. However, in some applications we would like to monitor a bandwidth large enough that sampling at the Nyquist rate is prohibitive or impossible, yet these applications feature sparse or compressible signals that can be compressively sampled using CS techniques. Intuitively, a sinusoid whose frequency and phase we are interested in measuring and tracking could have a very high frequency but is sparse in the Fourier basis with K near 1. Hence, such parameter estimation and tracking is a potential candidate for CS techniques.

Here we introduce a new family of digital and mixed analog/digital PLLs based on CS. Recall that the calculation used to update the phase estimate in the basic PLL of Figure 2.1 is the (weighted) inner product between the Nyquist rate samples $x[n]$ of the signal we wish to estimate/track and the estimated signal $u[n]$ that is generated by the oscillator. If both $x[n]$ and $u[n]$ can be represented by not only their Nyquist rate samples but also their (lower rate) compressive samples using compressive samplers

(with $x[n]$ producing $y[m]$ and $u[n]$ producing $v[m]$), then the RIP of CS guarantees that the standard inner product between their compressive samples $y[m]$ and $v[m]$ will be very close to the standard inner product between their Nyquist rate samples $x[n]$ and $y[n]$ (see Eq. 2.6).

Using this information, we introduce two compressive samplers into the basic PLL system (Figure 2.1) to create the CS-PLL, shown in Figure 3.1. The first acquires compressive samples $y[m]$ of the analog or discrete-time input signal $x(t)$. The second converts the oscillator's Nyquist rate samples $u[n]$ into compressive samples $v[m]$.

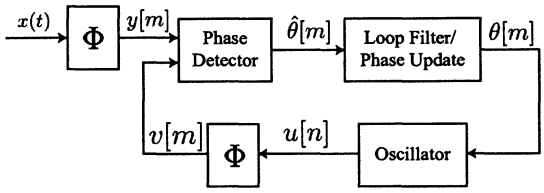


Figure 3.1: Block diagram of CS-PLL

The sampling operation in the loop should match and be synchronized with that used to sample the input. Due to the real-time nature of a PLL, this places some design constraints on the sampler, including casuality (we want to determine future phase information, not assume we know it already *priori*), low delay (to increase stability margin), and low system complexity. We note that the example compressive samplers described earlier (random demodulator, random sampler, and compressive multiplexer) satisfy these conditions and are reasonable to implement. For example, to implement the random demodulator in the loop, we input the same pseudo-random sequence to the two multipliers, use the same impulse response in the two integrators,

and synchronize the low-rate sampling in the two ADCs. In the case where $x(t)$ is an analog signal and $u[n]$ is a discrete-time signal, the behavior of the analog and digital compressive samplers should resemble one another. This may require some calibration, especially to limit the noise. All that needs to be stored for any digital implementation is the correct random seed for the pseudo-random sequence.

As with the classical PLL, the CS-PLL computes the phase estimate inner product using a multiplier and a filter. The filter can be non-linear and/or time varying. This filter often mimics the characteristics of the loop filter in the high-rate traditional PLL, yet with frequency scaling to adjust the filter bands for the new lower rate. The compressive sampler can be implemented in several different ways. For example it is possible to use a D/A converter to convert our phase estimate to analog, control a conventional hardware oscillator to generate a Nyquist rate signal $u(t)$, and then apply a hardware implementation of a compressive sampler to obtain $v[m]$. More simply, we model the compressive sampler digitally and use a linear time-invariant filter. The use of index m denotes a lower sampling rate than the Nyquist sampling counterpart.

$$\theta[m] = \sum_k y[k]v[k]h[m-k], \tag{3.1}$$

describes the phase estimate where the linear filter impulse response $h[m]$ acts as the kernel of the inner product, mimicing the impulse response of a higher rate filter in a traditional PLL. We note the parallel between Equations 2.1 and 3.1.

Chapter 4

Analysis

The fundamental principle of the CS-PLL, correlating a compressively sampled input and compressively sampled reference signal can be justified from an information theoretic perspective. Consider the minimum mean squared error (MMSE) and maximum likelihood (ML) estimators. We note that choosing a different objective function for optimization can result in different designs, which is fundamentally the basis of stochastic control. For example, creating a MAP estimator can lead to a PLL variant called the Costas loop [32]. We suppose that the system is frequency locked, and thus we are solely trying to estimate the phase of a signal [9]. (This is a safe assumption in a PLL system as well, as large frequency offsets prevent the loop from locking whereas small offsets are handled with such things as a second-order loop). If the phase is varying slowly enough over a set of M measurements, we can treat it as a single parameter ML estimation problem, and extend this framework to a PLL system where we adjust our phase estimate over time and filter unwanted noise in the loop.

If our signal of interest

$$x(t) = \cos(\omega_c t + \theta) \tag{4.1}$$

(with ω_c the continuous time frequency), is compressively sampled and corrupted with additive white Gaussian noise (AWGN) $w_i[m]$, we obtain

$$y[m] = \sum_{k=-\infty}^{\infty} p_m[k] \cos(\omega k + \theta) + w_i[m] \tag{4.2}$$

where $p_m[k]$ denotes the pseudo-random coefficients for sample m (essentially element (m, k) of sampling matrix Φ for samplers with a matrix formulation) and ω is the Nyquist-rate discrete-time frequency corresponding to ω_c. We note that AWGN could be added to $x(t)$ instead. Due to the characteristics of $p_m[k]$ as explained in Chapter 2.4 on noise amplification, AWGN added to $x(t)$ simply results in AWGN on $y[m]$ with higher variance, and thus for simplicity we assume the noise is added directly to $y[m]$. We assume a discrete set of possible times at which we can extract information, though by no means do we need to extract information at every time in our discretization.

For generality's sake, we use an infinite sum of random coefficients $p_m[k]$ for each sample m, but in practice most of the $p_m[k]$ would be zero. For example, the random demodulator has a window of length L based on the compression ratio where $p_m[k]$ would be ± 1. Although $p_m[k]$ is produced as a pseudorandom sequence, we treat it as deterministic over any set of M measurements since we know the values of the

sequence coefficients when peforming estimation. Rather our source of randomness is solely the input noise $w_i[m]$.

The tracking error variance of the unbiased estimator $\sum_{k=-\infty}^{\infty} p_m[k]\cos(\omega k + \tilde{\theta})$ assuming each measurement is independent is

$$\sigma^2 = \sum_{m=1}^{M} \left(y[m] - \sum_{k=-\infty}^{\infty} p_m[k]\cos(\omega k + \tilde{\theta}) \right)^2 \tag{4.3}$$

where $\tilde{\theta}$ represents the quantity we are estimating.

To find the MMSE estimator, we would like to minimize this quantity. Similarly if we assume that any noise added to $y[m]$ is uncorrelated with equal variance, we can determine a maximum likelihood estimate. For M samples of a signal with unknown phase, we would like to maximize the probablity density $p(y|\theta) = (2\pi)^{-\frac{M}{2}}|R^{-1/2}|e^{-\frac{1}{2}[y-v]^T R^{-1}[y-v]}$ where y and v are the input and estimate signals (and hence v is a function of $\tilde{\theta}$ defined as $\sum_{k=-\infty}^{\infty} p_m[k]\cos(\omega k+\tilde{\theta})$. Taking the log for the log likelihood ratio, we get $L(\theta|y) = -\frac{M}{2}\log(2\pi) - \frac{1}{2}\log(|R|) - \frac{1}{2}[y-v]^T R^{-1}[y-v]$. Letting R be $\sigma^2 I$ as we assumed, $L(\theta|y) = -\frac{M}{2}\log(2\pi) - \frac{1}{2}\log(M\sigma^2) - \frac{1}{2\sigma^2}[y-v]^T[y-v]$. Unsurprisingly after taking the derivative with respect to $\tilde{\theta}$ to maximize, we find this is equivalent to the MMSE estimation problem due to our assumption of independent measurements.

Taking the derivative of σ^2 with respect to $\tilde{\theta}$

$$\frac{\partial \sigma^2}{\partial \tilde{\theta}} = \frac{\partial}{\partial \tilde{\theta}} \sum_{m=1}^{M} \left(y[m] - \sum_{k=-\infty}^{\infty} p_m[k] \cos(\omega k + \tilde{\theta}) \right)^2 \tag{4.4}$$

$$= 2 \sum_{m=1}^{M} \left(\left(y[m] - \sum_{k=-\infty}^{\infty} p_m[k] \cos(\omega k + \tilde{\theta}) \right) \left(\sum_{k=-\infty}^{\infty} p_m[k] \sin(\omega k + \tilde{\theta}) \right) \right) \tag{4.5}$$

$$= 2 \sum_{m=1}^{M} \left(y[m] \sum_{k=-\infty}^{\infty} p_m[k] \sin(\omega k + \tilde{\theta}) \right.$$
$$\left. - \left(\sum_{k=-\infty}^{\infty} p_m[k] \cos(\omega k + \tilde{\theta}) \right) \left(\sum_{k=-\infty}^{\infty} p_m[k] \sin(\omega k + \tilde{\theta}) \right) \right) \tag{4.6}$$

To optimize the objective function, we need to set Eq. 4.6 to zero. We see two terms present for each measurement m, a correlation of $y[m]$ with compressive measurements of an 90-degree out-of-phase reference output, and a offset term independent of our output. The correlation term confirms that the compressive sampler model in the loop should match that used on the input, as adding or dropping terms increases the noise. For example, while it is viable to set some of the $p_m[k]$ to zero in the model, this is not optimal from an ML estimation perspective. We explore this offset term further. We notice that if the $p_m[k]$ terms are generated independently of each other with mean zero, when we treat $p_m[k]$ as random and consider the average case of M samples by finding the expected value over $p_m[k]$. the second terms in the summation reduce to $\sum_{k=-\infty}^{\infty} (p_m[k])^2 \cos(2(\omega k + \tilde{\theta}))$, which would be filtered out by a low-pass filter (conveniently present in a PLL). This is important because although we can continue to take measurements to reduce the variance of our estimate in a single parameter ML estimation, in a PLL the phase parameter is varying over time

and thus after a finite number of measurements we are now estimating a new phase value. For the next block of measurements we have a different (though known) $p_m[k]$ sequence, and thus by averaging over these sequences we can determine not only the effect of noise on our system, but find an average case performance. Furthermore the PLL naturally acts to achieve average case performance over time with its loop filters and adaptive nature.

We could surmise that one potential implementation of a phase-detector in a phase-locked loop is to correlate a compressive sampled sine input with a compressively sampled cosine input, and include the offset

$$-\left(\sum_{k=-\infty}^{\infty} p_m[k]\cos(\omega k+\tilde{\theta})\right)\left(\sum_{k=-\infty}^{\infty} p_m[k]\sin(\omega k+\tilde{\theta})\right)$$

at each iteration determined by our current estimate of the angle. Since we are treating the error independently among the M measurements, in a streaming PLL with varying phase we consider the case where $M = 1$ and determine the offset term at each iteration Given that:

- the offset term has an expected value of zero over the set of pseudo-random sequences (after filtering the terms at twice the signal frequency)
- filtering is performed in a PLL to reduce noise
- the original assumption was that the frequency was locked and we were solely estimating a non-varying phase parameter, which is not true in scenerios such

as FM

we tested systems with and without this offset included in the phase detector, and see that the bias minimally improves performance, especially at more moderate input SNRs. Comparison performance plots are included in the simulations section (Fig 7.7). Furthermore, ignoring this bias simplifies our system's complexity and corrresponding analysis model, and thus we ignore this correction in most of the work.

In the case of a complex PLL, the offset term completely cancels. For example, if we suppose our input signal is

$$x(t) = e^{j(\omega_c t+\theta)} \tag{4.7}$$

we would like to minimize

$$\sigma^2 = \sum_{m=1}^{M} \left| y[m] - \sum_{k=-\infty}^{\infty} p_m[k] e^{j(\omega k+\theta)} \right|^2 \tag{4.8}$$

Using Euler's identity to expand and writing $y[m]$ as $r[m](\cos(\psi[m])+j\sin(\psi[m]))$, we write σ^2 as

$$\begin{aligned} \sigma^2 = \sum_{m=1}^{M} \Bigg(& \left(r[m]\cos(\psi[m]) - \sum_{k=-\infty}^{\infty} p_m[k]\cos(\omega k+\theta) \right)^2 \\ & + \left(r[m]\sin(\psi[m]) - \sum_{k=-\infty}^{\infty} p_m[k]\sin(\omega k+\theta) \right)^2 \Bigg) \end{aligned} \tag{4.9}$$

Taking the derivative and setting to zero, we find that we get positive and negative

$\left(\sum_{k=-\infty}^{\infty} p_m[k]\sin(\omega k+\theta)\right)\left(\sum_{k=-\infty}^{\infty} p_m[k]\cos(\omega k+\theta)\right)$ terms that cancel and are left with

$$2r[m]\left(\cos(\psi[m])\sum_{k=-\infty}^{\infty} p_m[k]\sin(\omega k+\theta)-\sin(\psi[m])\sum_{k=-\infty}^{\infty} p_m[k]\cos(\omega k+\theta)\right)=0 \tag{4.10}$$

indicating that we should minimize the correlation between real and imaginary parts of our signal and reference.

A Cramer-Rao bound can be computed for the estimator. First taking the derivative of the log-likelihood ratio as before (and including a constant factor of $-\frac{1}{2\sigma^2}$ not present in the variance),

$$\frac{\partial L}{\partial\tilde{\theta}}=\frac{\partial}{\partial\tilde{\theta}}\left[-\frac{1}{2\sigma^2}\sum_{m=1}^{M}\left(y[m]-\sum_{k=-\infty}^{\infty} p_m[k]\cos(\omega k+\tilde{\theta})\right)^2\right] \tag{4.11}$$

$$\begin{aligned}=-\frac{1}{2\sigma^2}2\sum_m\Bigg(&y[m]\sum_{k=-\infty}^{\infty} p_m[k]\sin(\omega k+\tilde{\theta})\\&-\left(\sum_{k=-\infty}^{\infty} p_m[k]\cos(\omega k+\tilde{\theta})\right)\left(\sum_{k=-\infty}^{\infty} p_m[k]\sin(\omega k+\tilde{\theta})\right)\Bigg)\end{aligned} \tag{4.12}$$

Next,

$$\frac{\partial^2 L}{\partial \tilde{\theta}^2} = -\frac{1}{2\sigma^2} 2 \sum_{m=1}^{M} \left(y[m] \sum_{k=-\infty}^{\infty} p_m[k] \cos(\omega k + \tilde{\theta}) \right.$$
$$- \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} p_m[k] p_m[l] \cos(\omega k + \tilde{\theta}) \cos(\omega l + \tilde{\theta})$$
$$\left. + \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} p_m[k] p_m[l] \sin(\omega k + \tilde{\theta}) \sin(\omega l + \tilde{\theta}) \right) \tag{4.13}$$

Since

$$E_{w_i}[y] = \sum_{k=-\infty}^{\infty} p_m[k] \cos(\omega k + \tilde{\theta}) \tag{4.14}$$

then

$$-E_{w_i}\left[\frac{\partial^2 L}{\partial \tilde{\theta}^2}\right] = \frac{1}{2\sigma^2} 2 \sum_{m=1}^{M} \left(\sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} p_m[k] p_m[l] \sin(\omega k + \tilde{\theta}) \sin(\omega l + \tilde{\theta}) \right) \tag{4.15}$$
$$= \frac{1}{2\sigma^2} 2 \sum_{m=1}^{M} \left(\sum_{k=-\infty}^{\infty} (p_m[k])^2 \left(\frac{1}{2} - \frac{1}{2} \cos(2(\omega l - \tilde{\theta})) \right) \right.$$
$$\left. + \sum_{\substack{k=-\infty \\ k \neq l}}^{\infty} \sum_{l=-\infty}^{\infty} p_m[k] p_m[l] \sin(\omega k + \tilde{\theta}) \sin(\omega l + \tilde{\theta}) \right) \tag{4.16}$$

We use the notation $E_{w_i}[\cdot]$ to clarify that the expected value is taken with respect to noise $w_i[n]$, not $p_m[k]$. If we again consider the average case over time and treat $p_m[k]$ as zero-mean independent terms as well, then last set of terms will average to 0. Furthermore if $p_m[k]$ have unit average norm and we ignore the high frequency

terms, the Fisher information $F(\theta)$ reduces to

$$F(\theta) \approx \frac{M}{2\sigma^2}. \tag{4.17}$$

yielding and average Cramer-Rao bound of

$$\frac{2\sigma^2}{M}. \tag{4.18}$$

We see from the previous analysis that a Euclidean inner product of zero between the signal and estimate over a window of time nearly minimizes our estimation error. This follows very closely with the framework for a traditional PLL which tries to minimize the correlation between a Nyquist sampled input signal and Nyquist sampled sinusoidal estimate. As discussed in Chapter 2.2, the Euclidean inner products are approximately preserved in CS. The inner product of two compressively sensed vectors differs from the inner product of those two vectors by at most the product of the norms of those two vectors times a small constant, as detailed by Eq. 2.7.

To extend this principle to a streaming signal, we first consider an open-loop architecture such as a lock-in amplifier with a moving average filter. (Lock-in amplifiers contain a phase detector and low-pass filter, similar to a PLL, but the reference signal can be provided externally rather than through feedback.) We assume that the reference signal is rather close in frequency to the input signal. Suppose we compare a Nyquist rate system with a moving average filter to a corresponding compressive

system with moving average filter with length reduced by the compression factor (as per the reduction in sampling rate). We can apply the preservation of inner products bounds such as Eq. 2.7 directly to these systems. Each is trying to estimate the phase over the filter length, with the length of the filter corresponding to the number of CS measurements used at the instantaneous sample time.

We would like the assumption that the inner product is approximately preserved to hold over all time. The two inputs to the phase detector will have roughly B/f_s percentage of the spectrum full, implying we will be taking the inner product of two signals with frequency content in a bandwidth B over the full high-rate spectrum. If a modulating function besides sinusoids is used, such as square waves, more frequency components are present in our input and thus we would require a larger number of measurements. From classical CS theory to maintain the RIP of the sampling matrix and preserve the inner product, we would require $O(B \log f_s/B)$ measurements, with a constant multiplier of roughly 2 in practice when using the natural log. And given that we are often largely oversampled in the PLL, this number will be quite low. Adjusting the loop bandwidth to handle larger center frequency changes reduces the allowable compressive sampling rate only slightly more than linearly.

However, even the lock-in amplifier assumes that the the phase parameter stays relatively constant over the window of sampling. If we wish to track a varying phase as in the case of the PLL, we need to add a feedback loop to adjust the reference input. Although we often consider the loop filter and phase update as a

single component, the filter with response $H(z)$ can be broken into multiple filters $H(z) = H_1(z)H_2(z)\dots H_n(z)$. If $H_1(z)$ ia a moving average filter, we can apply the same CS bounds as before to the output of $H_1(z)$, and the error will be propagated through the remaining filters $H_2(z)$ through $H_n(z)$. The length of filter $H_1(n)$ again determines the size of our CS sampling matrix.

If we assume $H_1(z)$ is a moving average filter, worst case bounds can be found by assuming that the generated error at the output of $H_1(z)$ is a low frequency signal and simply scaled by any remaining loop filters $H_2(z)$ to $H_n(z)$. Assuming a traditional PLL is locked, we set $\langle x, u\rangle = 0$ to determine the deviation in the compressive inner product from zero, as $|\langle \Phi x, \Phi u\rangle| \leq \delta \|x\|_2 \|u\|_2$. Due to the randomness in the compressive samplers, linear or nonlinear analysis can be used to propagate this error, as it is independent of the input. To simplify the error estimate even further we can assume that $\|x\|_2 \leq \sqrt{M}$ and $\|u\|_2 \leq \sqrt{M}$ for unit amplitide sinusoids with M samples. However the results of this analysis are quite loose, both due to the assumption that the error signal is entirely low frequency and thus all its energy passes through the remaining filters $H_2(z)$ through $H_n(z)$ as well as the extremely loose bound on $\|x\|_2$ and $\|u\|_2$ for most samples. To generate a system of equations describing the effects of this error we assume the noise and input are independent. With zero signal input, our transfer function for the phase to the error becomes

$$\frac{\theta(z)}{I(z)} = \frac{H_2(z)\dots H_n(z)}{z - 1 + H_1(z)H_2(z)\dots H_n(z)} \tag{4.19}$$

where is $I(z)$ is a noise error input after the first moving filter $H_1(z)$. In this case, because our input $I(z)$ is a noise error, $\theta(z)$ represents a phase error quantity. For a constant input, $I(z) = \delta\frac{z}{z-1}$, we find that the steady state error e_{ss}

$$e_{ss} = \lim_{z\to 1} -(z-1)\frac{\theta(z)}{I(z)} = -\frac{\delta}{M_{sum}} \tag{4.20}$$

where M_{sum} represents the sum of the moving average taps. Thus for a constant error after the moving average filter, we find that a non-zero steady state error results. As we will see later however, rather than appearing low-frequency, the resulting noise in practice actually resembles Gaussian and is mostly removed from our system.

Unfortunately, using a large moving average filter increases the delay in the loop, reducing the stability margin of our PLL. Rather, PLLs are frequently designed with low-order, low-pass IIR filters. If we attempt to integrate the moving average filter characteristics into a low order filter (*ie* design $H_2(z)$ such that $H(z) = H_1(z)H_2(z)$ is a low order IIR filter), we find that $H_2(z)$ is only marginally stable with poles around the unit circle to replace the zeros of the moving average filter.

An alternative is to reduce the length of the moving average filter to increase the stability margin by removing delay. In the limiting case as we reduce the moving average filter length to 1, the phase detector is solely an inner product of a single measurement, which we would compare against the inner product of N/M measurements at the Nyquist-rate (equivalent to a CS sampling matrix with a single row). We do indeed realize that the bound on the error of the compressive inner product is

no longer particularly strong (large δ in Eq. 2.7) as we are not averaging noise over many measurements.

Suppose for a second that we consider the loop filter as a weighting of samples from the phase detector in an attempt to provide a better bound on the error in the compressive measurements. We now have a weighted inner product. Unfortunately weighted inner products are not preserved in general. To establish this, consider that this would require $E(\|\Phi x\|_{W_M}^2) = \|x\|_{W_N}^2 \forall x \in \mathbb{R}^N$, where diagonal matrices W_N and W_M represent the weighting of the norms at the high and low rates [2]. Now, consider the following extreme case: suppose x is 1-sparse signal and W_N is a weighting such that a single element with index k of x has weight $\epsilon > 0$ while all other elements have weight 1. (We require $\epsilon > 0$ to ensure positive definiteness of the inner product. If $\epsilon \ll 1$, and x_k is the non-zero element, the inner product will be particularly small. But since Φ is a random matrix, and an energy spreading operation, in expectation enough energy will be spread to at least one other element. Hence these would all need to be weighted very low in W_M, on the order of ϵ. On the other hand, if x_k is zero, one of the other elements has a lot of energy from W_N. But if we then spread all the energy using Φ, and weight W_M such that all elements except x_k are weighted low, then we lose too much energy from this transformation, yielding a contradiction. Alternatively, if $\epsilon \gg 1$ such that a single element is weighted more heavily, we arrive at a similar contradiction. Either Φ disperses the energy too much if the single element is weighted largely from the beginning, or we obtain too much energy as the

non-zero element is one of the original elements weighted 1. As we cannot establish that the weighted inner products concentrates around any weighted norm for all x, for general W_N, weighted inner products are not preserved. We could potentially minimize $\|W_M - \Phi^T W_N \Phi\|_2$ at each iteration of the loop over W_M (although W_N is non-time varying, Φ is time varying) such that W_M is diagonal to define a new filter that would potentially mimic the characteristics of the original high-rate filters, but this has many disadvantages: it would be extremely computationally complex, reduce stability, and works only for FIR filters.

Fortunately, it is not necessary to preserve weighted inner products in general. We have a very specific scenerio in the PLL in that we only require an estimate that is completely decorrelated with the input. By allowing feedback, we can average out any deviations in the inner product from zero to allow the PLL to regain lock. While probabilistically, there are cases where the compressive inner product may differ greatly from the original inner product, the effects on a PLL are much less substantial. Also, the effects of a single poor measurement quickly diminish as a measurement slides after the first few taps of a low-pass IIR filter.

Unlike CS reconstruction algorithms that rely on optimization, we do not have the opportunity to correct errors of previous iterations at the next time step (we can only affect future outputs), and must consider the stability and other characteristics of our system. Given that we have not yet ensured that the system remains stable and locked, we now consider a different tact, evaluating the system from a Nyquist

model framework. Several of the major characteristics of traditional PLL that can be evaluated using a simple linear model are:

1. stability
2. bandwidth
3. loop delay
4. steady state phase error

It should be noted that under some conditions, such as extremely low SNR, it is possible that the linear characteristics do not hold. However, the approximation is sufficient for many expected operating conditions, especially given that CS in general exhibits poor performance in the high noise regime.

On the other hand, several characteristics require non-linear analysis:

1. lock time
2. cycle slipping
3. instantaneous phase error

Non-linear analysis is difficult even in simple cases without randomization in the sampler, and often requires detailed simulations that are not useful for an understanding of the fundamental workings of a system. Rather for non-linear systems, the most efficient technique for analysis is to linearize the system. We utilize this approach, and will reserve nonlinear analysis of the CS-PLL for future work.

4.1 Nonlinear Model for the CS-PLL

To analyze the CS-PLL we first model the system in a simpler fashion comparable to a traditional PLL with an implicit downsampler and noise term. We consider systems with delay here, such as the random demodulator and interleaved demodulators. For systems without delay, such as random sampling and CMUX, many of the characteristics are similar to a PLL operating at the lower rate, the difference being an additional/different form of noise in loop. If we assume that the compressive sampler computes measurements in blocks using either Gaussian or Bernoulli sequences with unit norm per row, then we obtain zero-mean error in the phase detector. Given a discretized version of the random demodulator with chipping sequence taps p_m $\left[\frac{N}{M}m + k\right]$ and Nyquist rate samples $x[k] = \sin(\omega k + \theta_1[k])$ and $u[l] = \cos(\omega l + \theta_2[l])$ with ω the discrete-time Nyquist frequency, the multiplier creates

$$\hat{\theta}[m] = \left(\sum_{k=0}^{L-1} p_m\left[\frac{N}{M}m+k\right] x\left[\frac{N}{M}m+k\right]\right)\left(\sum_{l=0}^{L-1} p_m\left[\frac{N}{M}m+l\right] u\left[\frac{N}{M}m+l\right]\right) \tag{4.21}$$

$$= \sum_{k=0}^{L-1}\left(p_m\left[\frac{N}{M}m+k\right]\right)^2 x\left[\frac{N}{M}m+k\right] u\left[\frac{N}{M}m+k\right] + \sum_{\substack{k=0\\k\neq l}}^{L-1}\sum_{l=0}^{L-1} p_m\left[\frac{N}{M}m+k\right] x\left[\frac{N}{M}m+k\right] p_m\left[\frac{N}{M}m+l\right] u\left[\frac{N}{M}m+l\right] \tag{4.22}$$

$$= \sum_{k=0}^{L-1}\sin\left(\omega\left(\frac{N}{M}m+k\right)+\theta_1[k]\right)\cos\left(\omega\left(\frac{N}{M}m+l\right)+\theta_2[l]\right) + \sum_{\substack{k=0\\k\neq l}}^{L-1}\sum_{l=0}^{L-1}\left(p_m\left[\frac{N}{M}m+k\right] p_m\left[\frac{N}{M}m+l\right] \times \sin\left(\omega\left(\frac{N}{M}m+k\right)+\theta_1[k]\right)\cos\left(\omega\left(\frac{N}{M}m+l\right)+\theta_2[l]\right)\right) \tag{4.23}$$

where L is the length of the random demodulator band. Since $p_m\left[\frac{N}{M}m+k\right]$ for the random demodulator with Bernoulli taps are ± 1, $\left(p_m\left[\frac{N}{M}m+k\right]\right)^2 = 1$ and thus do not appear in Eq. 4.23.

The first summation in Eq. 4.23 accumulates Nyquist-rate samples. However, the oscillator updates only once per low-rate sample. Hence we model the system with an accumulate-and-dump after the phase detector, and a sample-and-hold before the oscillator.

The second summation in Eq. 4.23 represents cross-term noise $w_c[m]$. Because it is added to the low-rate phase estimate, not to individual high-rate Nyquist samples,

we model it as a noise term appearing after the accumulate and dump. Alternatively in the case of a single (non-interleaved) random demodulator, we can shift the noise term before the downsampler or accumulator with the appropriate scaling (letting noise accumulate as well) for modelling purposes; the models would be equivalent because the sample-and-hold elements prevent noise from affecting the oscillator except at the lower rate. However, from a conceptual standpoint, since the noise is a collection of cross-terms at the low-rate, modelling the noise after the downsampler is most consistent.

In fact, this cross-term noise will be both zero-mean and uncorrelated with the input (and feedback) due to the randomness introduced by the random demodulator. Jumping temporarily to matrix/vector notation for simplicity to compute the expected value and variance of our error estimate, we define x, u, and p to be $L \times 1$ vectors consisting of samples $x\left[\frac{N}{M}\right]$ to $x\left[\frac{N}{M} + (L-1)\right]$, $u\left[\frac{N}{M}\right]$ to $u\left[\frac{N}{M} + (L-1)\right]$, and $p_m\left[\frac{N}{M}\right]$ to $p_m\left[\frac{N}{M} + (L-1)\right]$ respectively and denote our phase error estimate $\hat{\theta}[m]$ for sample m simply as $\hat{\theta}$.

Suppose that $\hat{\theta} = \langle x, p\rangle \cdot \langle u, p\rangle$. Then we can write

$$\hat{\theta} = x^T p p^T u.$$

From the linearity of expectation we have that

$$E\left[\hat{\theta}\right] = E\left[x^T p p^T u\right] = x^T E\left[p p^T\right] u.$$

Since the entries of p are chosen independently and with variance 1, we have that $E\left[pp^T\right] = I$ (where I denotes the identity matrix), and thus $E\left[\hat{\theta}\right] = x^T u = \langle x, u \rangle$, as desired.

We now consider the variance

$$\begin{aligned}
\mathrm{Var}\left(\hat{\theta}\right) &= E\left[\hat{\theta}^2\right] - \left(E\left[\hat{\theta}\right]\right)^2 \\
&= E\left[u^T pp^T xx^T pp^T u\right] - u^T xx^T u \\
&= u^T E\left[pp^T xx^T pp^T\right] u - u^T xx^T u.
\end{aligned}$$

Let $\gamma = pp^T x$, and note that this is an $L \times 1$ vector with entries given by $\gamma_i = p_i \sum_{k=1}^{L} p_k x_k$. We next note that

$$\begin{aligned}
E\left[\gamma_i \gamma_j\right] &= E\left[\left(p_i \sum_{k=1}^{L} p_k x_k\right)\left(p_j \sum_{l=1}^{L} p_l x_l\right)\right] \\
&= \sum_{k=1}^{L}\sum_{l=1}^{L} E\left[p_i p_j p_k p_l x_k x_l\right]
\end{aligned}$$

Splitting this into two cases, we observe that when $i \neq j$, $E\left[p_i p_j p_k p_l x_k x_l\right] \neq 0$ only when $k = i$ and $l = j$ or $k = j$ and $l = i$. Thus, we obtain that for $i \neq j$, $E\left[\gamma_i \gamma_j\right] = 2x_i x_j$. In the case where $i = j$, we will have $E\left[p_i p_j p_k p_l x_k x_l\right] \neq 0$ whenever $k = l$, so that $E\left[\gamma_i \gamma_i\right] = \|x\|_2^2$. Combining these, we can write

$$E\left[\gamma\gamma^T\right] = D + 2xx^T$$

where D is a diagonal matrix with entries given by $d_{ii} = \|x\|_2^2 - 2x_i^2$.

Since $E\left[pp^T xx^T pp^T\right] = E\left[\gamma\gamma^T\right]$, we obtain

$$\mathrm{Var}\left(\hat{\theta}\right) = u^T(D + 2xx^T)u - u^T xx^T u = u^T Du + u^T xx^T u.$$

Note that

$$\begin{aligned} u^T Du &= \sum_{i=1}^{L} u_i^2(\|x\|_2^2 - 2x_i^2) \\ &= \|u\|_2^2 \|x\|_2^2 - 2\sum_{i=1}^{L} u_i^2 x_i^2 \end{aligned}$$

This simplifies to

$$\mathrm{Var}\left(\hat{\theta}\right) = |\langle x, u\rangle|^2 + \|x\|_2^2 \|u\|_2^2 - 2\sum_{i=1}^{L} u_i^2 x_i^2.$$

In many cases we can ignore $|\langle x, u\rangle|^2$ with x and u approximately orthogonal when a PLL is locked, but the Cauchy-Schwartz inequality provides us with an upper bound of

$$\mathrm{Var}\left(\hat{\theta}\right) \leq 2\|x\|_2^2 \|u\|_2^2 - 2\sum_{i=1}^{L} u_i^2 x_i^2 = 2\sum_{k\neq l} x_k^2 u_l^2.$$

Hence the variance clearly depends on the amount of subsampling. We could proceed to substite in $\sin(\omega k + \theta_k)$ and $\cos(\omega l + \theta_l)$ for x_k and u_l and apply trigonometric

identities, reducing the variance bound to

$$\mathrm{Var}\left(\hat{\theta}\right) \leq \sum_{k \neq l} \left(\frac{1}{2} \left(\sin(\omega(k-l) + \theta_k - \theta_l) + \sin(\omega(k+l) + \theta_k + \theta_l) \right)^2 \right).$$

If we add AWGN noise $w_i[n]$ to our input, this simply increases this upper bound. After squaring and applying further trigonometric identities, we see that some of the terms remain near baseband. Fortunately this is an upper bound. We do note that the variance quantity is time-varying.

Since the pseudo-random sequence has large enough periodicity, the ± 1 multipliers are essentially uncorrelated in the noise terms. We now resume indexing notation to show that if we assume $p_m\left[\frac{N}{M}m + k\right]$ and $p_m\left[\frac{N}{M}m + l\right]$ are Rademacher sequences, then a cross-term noise sample $w_c[m]$ is uncorrelated with any Nyquist input sample $x\left[\frac{N}{M}m + k\right]$ (where $w_c[m]$ is the collection of cross terms in the second summation of Eq. 4.23 as described above appearing as noise)

$$
\begin{aligned}
& E\left[x\left[\frac{N}{M}m+k\right]w_c[m]\right] \\
&= E\left[x\left[\frac{N}{M}m+k\right]\sum_{\substack{k=0\\k\neq l}}^{L-1}\sum_{l=0}^{L-1}x\left[\frac{N}{M}m+k\right]p_m\left[\frac{N}{M}m+l\right]u\left[\frac{N}{M}m+l\right]\right] \\
&= E\left[\sum_{\substack{k=0\\k\neq l}}^{L-1}\sum_{l=0}^{L-1}p_m\left[\frac{N}{M}m+k\right]p_m\left[\frac{N}{M}m+l\right]x^2\left[\frac{N}{M}m+k\right]u\left[\frac{N}{M}m+l\right]\right] \\
&= \sum_{\substack{k=0\\k\neq l}}^{L-1}\sum_{l=0}^{L-1}E\left[p_m\left[\frac{N}{M}m+k\right]\right]E\left[p_m\left[\frac{N}{M}m+l\right]\right]E\left[x^2\left[\frac{N}{M}m+k\right]u\left[\frac{N}{M}m+l\right]\right] \\
&= 0. \qquad (4.24)
\end{aligned}
$$

This same procedure can be repeated for all $x\left[\frac{N}{M}m+k\right]$ in the length L sequence, as well as similar analysis for the feedback. Any input outside the L-length sequence window is also uncorrelated with the noise, as the noise is solely a function of the pseudo-random sequence and terms in the L-length window.

We are not arguing that the noise is Gaussian with a fixed variance. However, we also note that for the application of a PLL all that we really need is approximately flat spectrum noise near baseband in the phase detector, as high frequency noise will be removed by the loop filter. This is a much less restrictive assumption than purely Gaussian noise.

We can in fact also show that the output of a system using a single random demodulator is equivalent to the output of a system approximately inverting the

application of a compressive sampler Φ by applying Φ^T before inputting the the PLL, and then using an accumulate-and-dump within the feedback loop after the phase detector. This shows the obvious connection to a Nyquist-rate PLL and the correlation between the noise in the loop and noise introduced on the input signal through the use of the Φ^T.

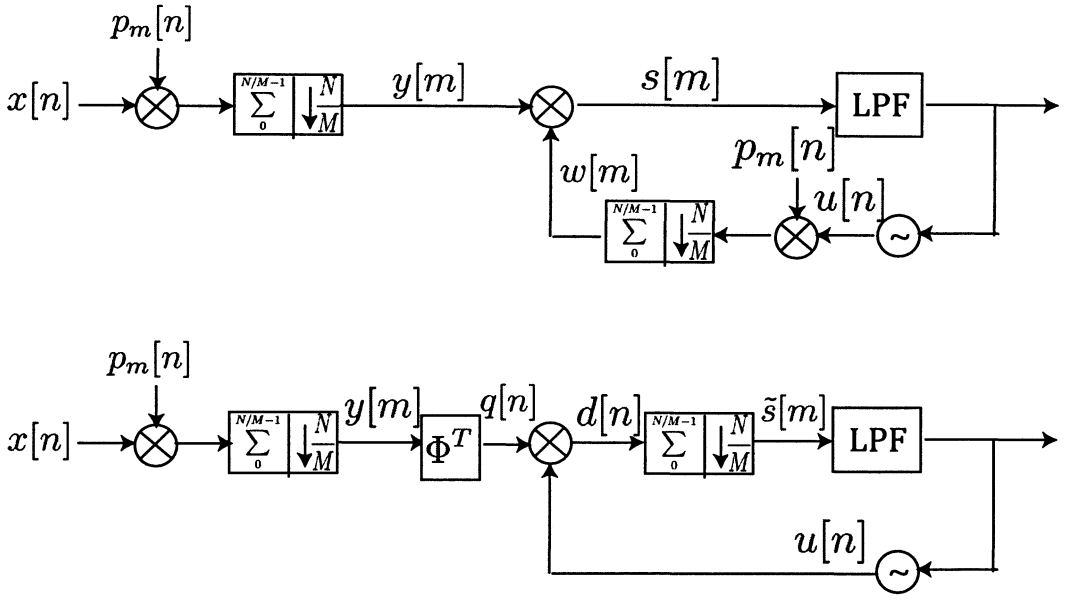


Figure 4.1: Diagram showing equivalent systems when using the random demodulator

We proceed as follows: consider the system diagrams in Figure 4.1. We need to show that the signals $s[m]$ and $\tilde{s}[m]$ are equal (we assume the same low-rate low-pass filter is used in either case). For Figure 4.1, we can write $s[m]$ as

$$s[m] = \left(\sum_{j=0}^{L-1} p_m\left[\frac{N}{M}+j\right] x\left[\frac{N}{M}+j\right]\right)\left(\sum_{i=0}^{L-1} p_m\left[\frac{N}{M}+i\right] u\left[\frac{N}{M}+i\right]\right) \tag{4.25}$$

$$= \sum_{j=0}^{L-1}(p_m\left[\frac{N}{M}+j\right])^2 x\left[\frac{N}{M}+j\right] u\left[\frac{N}{M}+j\right]$$

$$+ \sum_{j=0}^{L-1}\sum_{\substack{i=0 \\ i\neq j}}^{L-1} p_m\left[\frac{N}{M}+j\right] x\left[\frac{N}{M}+j\right] p_m\left[\frac{N}{M}+i\right] u\left[\frac{N}{M}+i\right] \tag{4.26}$$

The transpose operation for the linear demodulator is equivalent to an upsample by L followed by modulation by the pseudorandom sequence. Hence we denote $q[n]$ following the transpose operation as

$$q[n] = p_m[n]y[m] \tag{4.27}$$

$$= p_m[n]\left(\sum_{k=0}^{L-1} x\left[\frac{N}{M}+k\right] p_m\left[\frac{N}{M}+k\right]\right) \tag{4.28}$$

$$= \left(\sum_{\substack{k=0 \\ k\neq \bmod(n,L)}}^{L-1} x\left[\frac{N}{M}+k\right] p_m[n]\, p_m\left[\frac{N}{M}+k\right]\right)$$

$$+ (p_m\left[\frac{N}{M}+ \bmod(n,L)\right])^2 x\left[\frac{N}{M}+ \bmod(n,L)\right] \tag{4.29}$$

$$= \left(\sum_{\substack{k=0 \\ k\neq \bmod(n,L)}}^{L-1} x\left[\frac{N}{M}+k\right] p_m[n]\, p_m\left[\frac{N}{M}+k\right]\right) + (p_m[n])^2 x[n] \tag{4.30}$$

where $m = \lfloor \frac{n}{L} \rfloor$.

After the phase detector at each iteration we obtain the signal,

$$d[n] = q[n]u[n] \tag{4.31}$$

$$= \left(\sum_{\substack{k=0 \\ k \neq \bmod (n,L)}}^{L-1} x\left[\frac{N}{M}+k\right] p_m[n]\, p_m\left[\frac{N}{M}+k\right] u[n] \right) + (p_m[n])^2 x[n]\, u[n] \tag{4.32}$$

Lastly, after the accumulate and dump we get

$$\tilde{s}[m] = \sum_{b=0}^{L-1} q\left[\frac{N}{M}+b\right] u\left[\frac{N}{M}+b\right] \tag{4.33}$$

$$= \sum_{b=0}^{L-1} \left(\left(\sum_{\substack{k=0 \\ k \neq \bmod (n,L)}}^{L-1} x\left[\frac{N}{M}+k\right] p_m\left[\frac{N}{M}+b\right] p_m\left[\frac{N}{M}+k\right] u\left[\frac{N}{M}+b\right] \right) + (p_m\left[\frac{N}{M}+b\right])^2 x\left[\frac{N}{M}+b\right] u\left[\frac{N}{M}+b\right] \right) \tag{4.34}$$

which is an equivalent expression to Equation (4.26) after letting $i = k$ and $b = j$.

In the case of the interleaved demodulators, this is no longer true. We find that applying the transpose first spreads the weighting over more components of a Nyquist rate input at each iteration.

4.2 Linear Model

The CS-PLL can similarly be analyzed in a linear fashion. Because we have shown the noise is uncorrelated with the input, we apply the simple linearizing assumption

of the traditional PLL ($\sin(\theta) \approx \theta$).

If a linear model of the noise is required, we can linearize each multivariate expression using $\sin(x)\cos(y)$ as $\sin(x_0)\cos(y_0)+\cos(x_0)\cos(y_0)(x-x_0)-\sin(x_0)\sin(y_0)(y-y_0)$. However this is generally not necessary.

4.2.1 Classical PLL

We first outline the fundamental system characteristics of the linear model of a classical PLL to make the applicable connections to the CS-PLL. Without loss of generality, a second order loop as described by Eq. 2.3 will be used with the classical PLL as this is sufficient for tracking both frequency and phase changes, and often used in practice.

A simple stability analysis can be performed from the linear model. The poles of the linear transfer function,

$$H_{cl}(z) = \frac{C_2(z-1)+C_1}{(z-1)^2+C_2(z-1)+C_1} \tag{4.35}$$

must be in the unit circle, found by solving $(z-1)^2+C_2(z-1)+C_1 = 0$. Solving for C_1 and C_2, it is shown that $C_1 > 0$, $C_1 > 2C_2 - 4$ and $C_1 < C_2$ are the conditions required for stability [38,39].

Bandwidth of the PLL is determined simply by the loop filter. The loop filter will remove everything except low frequency noise (and aliasing). Wide bandwidth loop filters are useful for providing a larger range for lock whereas narrowband filters

help reduce wideband noise. However in-band noise continues to reduce output SNR. Loop Delay is also determined by the order of the filter. For a second-order system we have only a delay of 2.

Phase error can be computed either linearly or for more complicated analysis, nonlinearly. For a linear input phase change $\theta_i[n] = \delta_0 n$, the steady state error is $e_{ss} = \lim_{z\to 1} \frac{(z-1)^2 \delta_0 \frac{z}{(z-1)^2}}{C_2(1+\frac{1}{z-1}\frac{C_1}{C_2})+(z-1)} = 0$. The non-linear analysis requires resorting to Fokker-Planck even for simply a 2nd order loop filter with AWGN.

CS-PLL

The loop filter can similarly be designed for a CS-PLL to ensure a (linearly) stable closed loop. We can model the system as a high-rate PLL for the original Nyquist-sampled $x[n]$ with sample-and-hold elements in the feedback loop. Noise is added within the phase detector, consisting of all the cross term between sinusoidal samples modulated by the pseudorandom sequence (i.e.

$$\sum_{\substack{k=0 \\ k\neq l}}^{L-1} \sum_{l=0}^{L-1} p_m \left[\frac{N}{M}m + k\right] x \left[\frac{N}{M}m + k\right] p_m \left[\frac{N}{M}m + l\right] u \left[\frac{N}{M}m + l\right]$$

from (4.22)). AWGN could also be added to any component in the system, but does not directly affect the linear model formulation.

If we assume that the compressive sampler computes measurements in blocks using either Gaussian or Bernoulli sequences with unit norm per row, an FIR filter with L random taps equal to the $\left(p_m \left[\frac{N}{M}m + k\right]\right)^2$ terms follows the phase detector, and then

downsampling by $\frac{N}{M}$ is done. The FIR filter has all positive coefficients. The loop filter operates at the lower sampling rate and a sample and hold element is added to this input to the oscillator to return the sampling rate to the original high rate. The basic sample-and-hold model is shown in Fig. 4.2(a), while the linearized model is shown in Fig. 4.2(b). Cross-term noise is included as $w_c[m]$. Noise in the phase detector is ignored, and similar to [17], we can write the system equations. We start by finding the open loop function

$$H_{op} = \frac{\theta_o}{\theta_e}.$$

We develop the following set of z-domain equations for the linear model using a set of interleaved random demodulator with Bernoulli taps (for Gaussian taps $H_a(z)$ is the only component that changes; since the filter would be time-varying in this case,

the notation $H_a(z)$ is no longer entirely correct)

$$
\begin{aligned}
H_a(z) &= 1 + z^{-1} + z^{-2} + \ldots + z^{-(L-1)} \\
&= \frac{1 - z^{-L}}{1 - z^{-1}} \\
\hat{\theta}(z^{N/M}) &= \frac{1}{N/M} H_a(z)\theta_e(z) \\
\theta(z^{N/M}) &= \hat{\theta}(z^{N/M}) H_l(z^{N/M}) \\
H_b(z) &= 1 + z^{-1} + z^{-2} + \ldots + z^{-(N/M-1)} \\
&= \frac{1 - z^{-N/M}}{1 - z^{-1}} \\
\theta_f(z) &= \theta(z^{N/M}) H_b(z) \\
\theta_o(z) &= \theta_f(z) \frac{z^{-1}}{1 - z^{-1}} \\
\theta_e(z) &= \theta_i(z) - \theta_o(z)
\end{aligned}
$$

where $H_a(z)$ is the transfer function of the accumulator, $\hat{\theta}(z^{N/M})$ is the angle after downsampling, $H_l(z)$ is the loop filter, $\theta(z^{N/M})$ is the phase after the loop filter, $H_b(z)$ is the transfer function of the upsample and hold, $\theta_f(z)$ is the phase after upsample and hold, $\theta_o(z)$ is the phase after updating from the previous angle, $\theta_e(z)$ is the phase error, N/M is the compression ratio, and L is the band length in the compressive sampler. We assume any other loop gains from sources such as the oscillator or phase detector are compensated for by the proportional gain constant in the loop filter. We also assume there is no additional delay in the loop besides that introduced by the sample and hold components and the single sample delay required to update the

phase, if this is not true an additional z^{-D} term can be added included in the loop for D samples delay.

Continuing on, we find that

$$\begin{aligned} H_{op} &= \frac{\theta_o}{\theta_e} \\ &= \frac{z^{-1}}{1-z^{-1}} \frac{1}{N/M} \left(\frac{1-z^{-N/M}}{1-z^{-1}} \right) \left(\frac{1-z^{-L}}{1-z^{-1}} \right) H_l(z^{N/M}) \end{aligned}$$

and the closed loop transfer function can be formulated from this as

$$H_{cl} = \frac{H_{op}}{1+H_{op}}.$$

.

If we use the loop filter described by Eq. 2.3, we need to choose filter parameters C_1 and C_2 such that the poles of the transfer function

$$H_{cl}(z) = \frac{C_2 z^{-1}(1-z^{-N/M}(1-\frac{C_1}{C_2}))(1+\cdots+z^{-(L-1)})}{(\frac{N}{M})(1-z^{-1})^2 + C_2 z^{-1}(1-z^{-N/M}(1-\frac{C_1}{C_2}))(1+\cdots+z^{-(L-1)})} \tag{4.36}$$

are within the unit circle. Although solving this high-order polynomial analytically for bounds is difficult, we find heuristically that reducing C_1 and C_2 by the compression rate (relative to values used in a corresponding stable traditional PLL) makes the system stable, at least for the case of a single random demodulator. As we interleave

more demodulators, we must decrease the gain in the loop further to ensure stability.

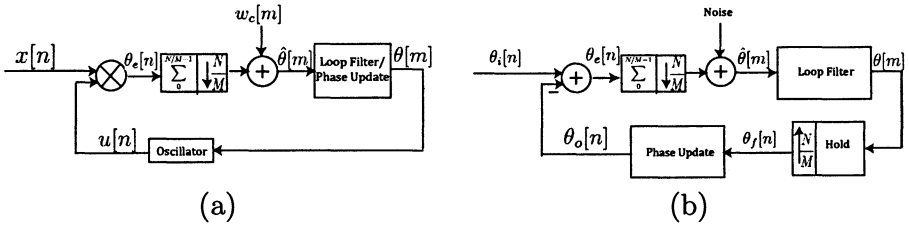


Figure 4.2: Sample-and-hold model for stability analysis using random demodulators. (a) Non-linear sample-and-hold model of CS-PLL. (b) Linear sample-and-hold model of CS-PLL.

The bandwidth of the PLL follows directly from the loop filter, however we remember that the system is now sampling at a lower rate in the loop. Because the noise introduced by the cross-terms is fairly wideband, narrowband loop filters can be particularly helpful after the loop has locked. An appropriate loop filter can be chosen to ensure that the linear transfer function $H_{cl}(z)$ has no poles outside the unit circle. Because of the accumulate-and-dump component in this model, the loop order is $L + N/M$, verifying the potential for increasing instability as the compression grows. L is always a multiple of N/M in the case of interleaved random demodulators, and as we increase the number of interleaved random demodulators, $L \gg N/M$, indicating the serious disadvantage of too many demodulators on stability.

As expected for a second order loop filter, the steady state phase error for a frequency offset is zero, found as before by solving

$$e_{ss} = \lim_{z \to 1}(z-1)(1 - H_{cl}(z)) \tag{4.37}$$

where $H_{cl}(z)$ is the closed loop transfer function calculated above.

We could perform additional analysis from the transfer function. For example, we can determine the sensitivity of the system to parameters such as the loop filter coefficients C_1 and C_2. If we consider the transfer function $H_{cl}(z) = N(z)/D(z)$, we can write the sensitivity to parameter $S_\beta^{H_{cl}}(z)$ as:

$$\begin{aligned} S_\beta^{H_{cl}}(z) &= \left.\frac{\Delta H_{cl}(z)/H_{cl}(z)}{\Delta\beta/\beta}\right|_{\beta_0} \\ &= \left.\frac{\beta}{H_{cl}(z)}\frac{\partial H_{cl}(z)}{\partial\beta}\right|_{\beta_0} \qquad (4.38) \\ &= \left.\left(\frac{\beta}{N(z)}\frac{\partial N(z)}{\partial\beta} - \frac{\beta}{D(z)}\frac{\partial D(z)}{\partial\beta}\right)\right|_{\beta_0} \qquad (4.39) \end{aligned}$$

after application of the chain rule.

Applying this for parameters C_1 and C_2 we get,

$$S_{C_1}^{H_{cl}}(z) = \frac{N_{C1}}{D_1 D_2} \qquad (4.40)$$

and

$$S_{C_2}^{H_{cl}}(z) = \frac{N_{C2}}{D_1 D_2}. \qquad (4.41)$$

where

$$N_{C1} = C_1 z^{-N/M-1}(1 + \cdots + z^{-(L-1)})(\frac{N}{M})(1 - z^{-1})^2 \tag{4.42}$$

$$N_{C2} = C_2 z^{-1}(1 - z^{-N/M})(1 + \cdots + z^{-(L-1)})(\frac{N}{M})(1 - z^{-1})^2 \tag{4.43}$$

$$D_1 = (\frac{N}{M})(1 - z^{-1})^2 + C_2 z^{-1}(1 - z^{-N/M}(1 - \frac{C_1}{C_2}))(1 + \cdots + z^{-(L-1)}) \tag{4.44}$$

$$D_2 = C_2 z^{-1}(1 - z^{-N/M}(1 - \frac{C_1}{C_2}))(1 + \cdots + z^{-(L-1)}) \tag{4.45}$$

$$\tag{4.46}$$

Example nominal values of C_1 and C_2 are 0.000149 and 0.024415 for a total spectral bandwidth of 2.048 MHz and loop bandwidth of 10 kHz.

We also note that the response of the PLL to Gaussian noise or other inputs can be approximated simply with transfer function, with $S_{out}(f) = |H_{cl}(f)|^2 S_{in}(f)$, where $S_{in}(f)$ and $S_{out}(f)$ represent the input and output power spectral densities.

4.3 Aliasing

As [17] points out, systems with an accumulate-and-dump in the loop can cause aliasing without removing noise in the wide bandwidth. However, what we realize is that CS already suffers from the noise folding issue. There is very little that can be done to avoid the drop in performance due to compression of input noise in the spectrum. Rather, CS would like to avoid the aliasing of large narrowband interference onto signals of interest. We see that because the Nyquist model for the

random demodulator includes an accumulate filter before the downsampling, noise is filtered. While moving average filters suffer from weak sidelobe suppression, in the case of the PLL, we particular concerned about the narrowband noise that would alias to baseband, signals at M/Nf_s intervals from the oscillator frequency in the original Nyquist rate spectrum. In fact, these are very sharp nulls of the moving average filter. For a single random demodulator, the nulls and regions of aliasing are equivalent; in the case of interleaved demodulators we find additional nulls in the spectrum beyond the regions of aliasing.

4.4 Gaussian Random Demodulators

As noted earlier, a random demodulator with Gaussian taps can also be used. In the case of the CS-PLL, this will imply that the accumulate-and-dump filter in our linear model is no longer a simple moving average filter, but rather the taps of the filter at each iteration would be the square of the demodulator taps. As this is a time-varying filter, an LTI system transfer function no longer exists. However the magnitude and phase response of this filter has many characteristics closely resembling that of a simple moving average. To ensure a consistent loop gain, the components could be normalized. In simulation we notice that this slightly affects performance. As compression increases, the norm of each row $\rightarrow 1$, such that see little difference between the normalized and unnormalized Gaussian coefficient cases. Using Gaussian taps affects the performance versus the use of the traditional random demodulator

with ± 1 taps in three ways:

- Cross-term noise is slightly lower
- Nulls are not quite as strong, allowing more energy from narrowband interferers to fold into the spectrum
- An average sampling position is varied

The first point is a direct result of the fact that equal variance between Gaussian and sub-Gaussian sampling schemes implies a lower variance of the sum for Gaussian terms. Weaker nulls would allow more energy from aliasing narrowband interferers to fold into the spectrum. The final point requires more explanation. Supposing that we have lock, our phase signal is very low in frequency. If we weight certain components of a linear combination of elements, we are in essence placing more emphasis on that component of the signal. If for example we have the filter taps [0.8 0.8 1.3 0.8 0.8 0.8] we see that we place particular emphasis on the third component. But if we then use taps, [1.3 0.8 0.8 0.8 0.8 0.8] at the next iteration preceding an integrate-and-dump filter, we essentially shifted our sampling position to the left, such that we now take a sample sooner than a uniform filter would expect. We see this structure exhibiting some characteristics similar to a random sampling configuration. Because the loop filter expects uniform sampling, these deviations appear as noise. One solution to improve performance would be to scale the loop gain based on an "average" sampling position. Instead of shifting by $\frac{N}{M}$ Nyquist-rate samples, we would scale the gain as

though we shifted by $\frac{N}{M}+\sum_{k=0}^{L-1}\frac{k}{L}\left(p_m\left[\frac{N}{M}m+k\right]-p_{m-1}\left[\frac{N}{M}(m-1)+k\right]\right)$ samples. Thus, we would apply an additional gain of

$$1+\frac{\sum_{k=0}^{L-1}\frac{k}{L}\left(p_m\left[\frac{N}{M}m+k\right]-p_{m-1}\left[\frac{N}{M}(m-1)+k\right]\right)}{\frac{N}{M}}.$$

Chapter 5

Kalman Filter

A simple second-order PLL can be formulated as a Kalman filter with time-invariant process statistics [47]. Here we first describe the traditional PLL formulation and then a similar formulation for a random demodulator as well as the limitations.

A Kalman filter is a linear model, though variants such as an extended Kalman filter could be used to handle non-linearities if necessary. In the traditional PLL case, we assume that that we have a two state vector consisting of phase and (discrete-time) frequency states.

$$x_m = \begin{bmatrix} \theta \\ \omega \end{bmatrix}$$

We directly observe the phase (since we assume a linear model), so our observed quantity z_m is

$$z_m = H_m x_m + v_m = \begin{bmatrix} 1 & 0 \end{bmatrix} x_m + v_m$$

where $v_m \sim \mathcal{N}(0, R_m)$.

We write the state update equations as

$$x_{m+1} = G_m x_m + w_m = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} x_m + w_m$$

where $w_m \sim \mathcal{N}(0, Q_m)$.

The predict and update equations denoting the Kalman filter can be written as

$$\hat{x}_{m|m-1} = G_m \hat{x}_{m-1|m-1} + D_m u_m$$

$$B_{m|m-1} = G_m B_{m-1|m-1} G_m^{\mathrm{T}} + Q_m$$

$$\tilde{y}_m = z_m - H_m \hat{x}_{m|m-1}$$

$$S_m = H_m B_{m|m-1} H_m^{\mathrm{T}} + R_m$$

$$K_m = B_{m|m-1} H_m^{\mathrm{T}} S_m^{-1}$$

$$\hat{x}_{m|m} = \hat{x}_{m|m-1} + K_m \tilde{y}_m$$

$$B_{m|m} = (I - K_m H_m) B_{m|m-1}$$

The control input u_m in this case is often 0.

We can simplify the Kalman filter gain K_m to

$$\begin{bmatrix} (B_{m|m-1})_{11} \\ (B_{m|m-1})_{21} \end{bmatrix} ((B_{m|m-1})_{11} + R_m)^{-1}$$

where $((B_{m|m-1})_{ij}$ denotes the (i,j)th element of $B_{m|m-1}$ and R_m is the measurement noise variance. Hence we see that the gain is primarily based on the predicted

covariance of the phase as expected.

We could develop a Kalman filter model for the CS-PLL using a random demodulator by using L phase states and 1 frequency state, where L again denotes the window length. We use the same set of equations but

$$x_m = \begin{bmatrix} \theta_0 \\ \vdots \\ \theta_{L-1} \\ \omega \end{bmatrix}$$

$$z_m = H_m x_m + v_m = \begin{bmatrix} p_m\left[\frac{N}{M}\right] & \dots & p_m\left[\frac{N}{M}+L-1\right] & 0 \end{bmatrix} x_m + v_m$$

with $p\left[\frac{N}{M}\right], \dots, p\left[\frac{N}{M}+L-1\right]$ the pseudo-random taps of the demodulator

and

$$x_{m+1} = G_m x_m + w_m = \begin{bmatrix} 1 & 0 & \dots & 0 & L \\ 0 & 1 & \dots & 0 & L \\ & & \ddots & & \\ 0 & 0 & \dots & 1 & L \\ 0 & 0 & \dots & 0 & 1 \end{bmatrix} x_m + w_m$$

Again ω denotes the Nyquist-rate discrete-time frequency, and H_m varies at each iteration unlike the traditional PLL where H_m is time-invariant. What is important to note, however, is that $B_{m|m}$ should indicate the strong correlation between states

θ_0 to θ_{L-1}. (We could also use multiple frequency states, tying ω_0 to θ_0, ω_1 to θ_1, etc. This assumes ω is changing within each block, and the strong correlation between ω_0 through ω_{L-1} would also need to be taken into account.) Assuming independence here is not at all true given that the phase values are measured all very close in time. Given that H_m is changing at each iteration and removing correlation among the phase values, the difficulty in formulating a valid correlation model, and the relative complexity of the computations, using a Kalman filter model is not nearly as feasible as the traditional PLL case. Interleaved random demodulators poses even more difficulties. The observation matrix assumes each set of observations is independent of the last, yet for example if we take 3 measurements of 6 states in an attempt to interleave 3 random demodulators, these 3 measurements would not be interleaved with the next set. Hence we would need to expand our matrix with states for the length of the signal, at which point we would be doing very large blockwise computation, still leaving us with transients at the end of each block and inordinate delay in the system. By introducing additional phase states, we would be introducing addtional delay into the feedback loop as the filtering of each large block of input is dependent upon the statistics of the previous large block.

Chapter 6

Additional Issues

The CS-PLL is applicable to many of the same signal processing problems where a traditional PLL is used on Nyquist-rate data. A block diagram showing a full layout of a sample radio architecture is shown in Figure 6.1. We outline several of the key practical considerations arising when integrating a CS-PLL system such as filtering, quantization, and pre- and de-emphasis.

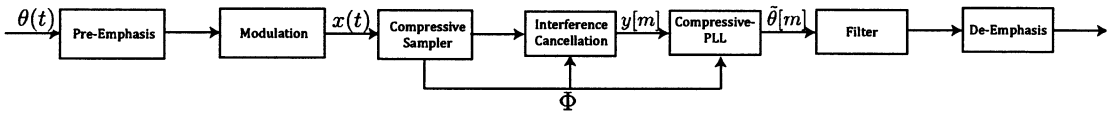


Figure 6.1: Block diagram of overall compressive FM receiver system

In [12], a simple compressive matched filter is used for detection, comparing the compressive measurements y to a set of compressively sampled sinusoids at discrete frequencies f_k. Often times this may simply be a uniformly spaced set of frequencies, but if we determine a detected signal is FM, then we could use these two components in conjunction. On the one hand, the result of our matched filtering could be provided as the output of the phase detector in a CS-PLL to reduce computation. On the other hand, we could update the discrete set of frequencies f_k for detection using the current

frequency output of the PLL to ensure that we continue tracking the FM signal and improve detection performance. This is particularly relevant because signals not on the Nyquist grid pose difficulties for many CS algorithms as they no longer appear sparse. If we assume that there will likely be only one signal in a particular portion of the spectrum, we could adjust one of our K detection frequencies f_k. Otherwise we could increase the size of our detection set to $K+1$, continuing to monitor the original K uniformly spaced frequencies as well. There is always obviously a tradeoff in terms of accuracy vs computation in the number of frequencies that we detect. However if we realize that an FM signal is dropped we could simply remove the frequency at which it was last detected from our set and resume matched filtering over the original discrete set.

6.1 Complex vs Real

There are tradeoffs to consider when choosing whether a PLL will be driven with either real or complex input for both the classical and CS-PLLs. Using complex data generally increases the SNR, while a real PLL may be easier to implement. In the ideal case we would like to work directly with complex data. However for the CS-PLL this implies taking compressive measurements of the complex FM signal. We cannot convert to complex data using a digital Hilbert transform after taking measurements; these operations do not commute. Even if complex data is used in a classical PLL, real data is actually obtained at the receiver. If we perform conversion to complex

FM data in analog hardware or obtain it through other means we can utilize the performance boost in a CS-PLL, though analog Hilbert transformers have their own drawbacks [21] .

In the high input SNR case, using real signals immediately drops the performance for the CS-PLL. Not only is there a large harmonic present at twice the carrier frequency as in the traditional PLL, but a moderate noise floor is immediately apparent that does not show up when using either a CS-PLL with complex data or a traditional PLL (with real or complex data). However as the input SNR drops, the complex CS-PLL and traditional PLLs develop noise floors as well.

Figure 6.2 shows a sample spectral output for the traditional and CS-PLLs, without any extra pre- or post-processing. The message signal is a sinusoidal tone at 2.5 kHz and the carrier frequency is 120 kHz in a bandwidth of 2.048 MHz, compressed by a factor of 2. For this example, no noise was added to the input. As the input SNR decreases, the advantage of the complex PLL will decrease. Both the harmonic at twice the center frequency for real data and higher noise floor for CS data are apparent from the figure.

6.2 Dynamic Range

Generally the effects of quantization will be overshadowed by either input noise or the nonlinear noise term from the compressive sampler. Hence we can model quantization noise as white noise with uniform probability uncorrelated with the

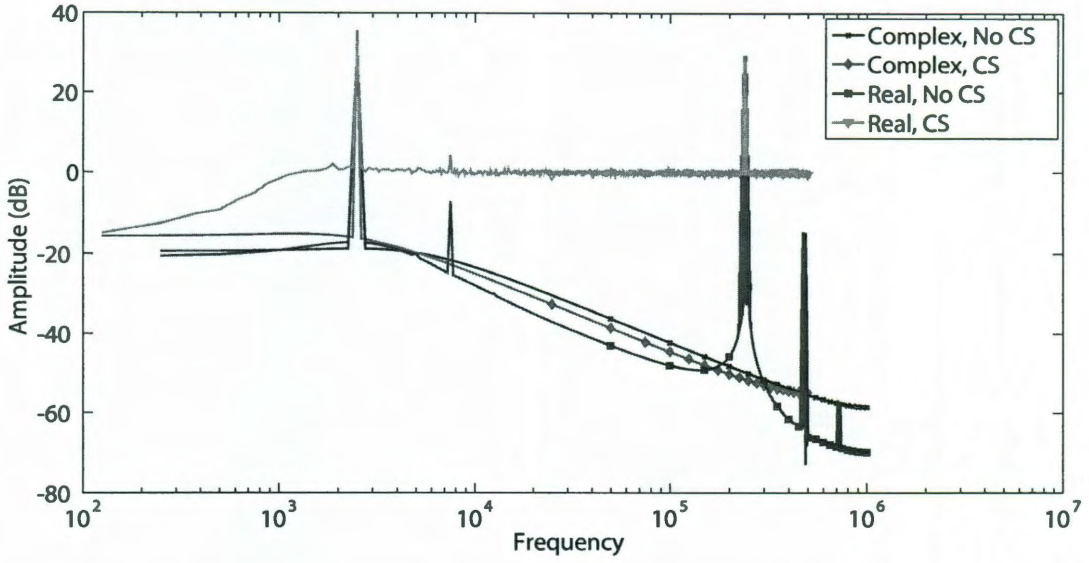


Figure 6.2: Sample spectral output of a traditional and CS-PLL

input and other noises in the system, and frequently ignore its impact due to its relative magnitude versus other noise factors. The CS-PLL exhibits similar dynamic range issues as the traditional PLL; the large harmonic present at twice the carrier frequency at the output of a CS-PLL operating on real data is generally approximately equal in amplitude to those seen in the output of the real traditional PLL, and the harmonic is generally far above the noise floor. This can be readily explained by the fact that while throughout much of the analysis we dropped the harmonic terms at twice the carrier frequency, each time a term of interest was present, an equal amplitude term at the harmonic frequency was generated from the trigonometric identity $\sin(\theta_1)\cos(\theta_2) = \frac{1}{2}\sin(\theta_1 - \theta_2) + \frac{1}{2}\sin(\theta_1 + \theta_2)$. Simulation results using a random demodulator confirmed that while the strength of the output centered at twice the carrier frequency varied with input SNR, it did not change as we varied

the quantization of our system. For example, with no noise on the input, the noise at twice the carrier frequency was 8 dB stronger than the baseband message signal, with 20 input SNR the noise was consistently 20 dB above the message signal, and at 8dB input SNR, the noise spike was 40 dB above the baseband message signal.

6.3 Democracy

In practice, CS measurements are quantized with a finite-range quantizer before they enter the CS-PLL. In some cases, the quantizer may saturate (i.e., acquiring measurements whose nominal value exceed the range $|T|$ of the quantizer will be thresholded to T), possibly resulting in large measurement error and hindering the ability of the CS-PLL to lock on the carrier frequency. We can utilize the democracy property of CS measurements to combat finite range measurements and still not impede the performance of the CS-PLL. The democracy property of CS measurement systems ensures that if we do not saturate too often, then the sampler maintains RIP and the angles are preserved [25]. The angle estimate $\widehat{\theta}[m]$ is computed as before, however, in a time-instance m where an input measurement exceeds the threshold, i.e., $y[m] > T$ or $y[m] < -T$, the state of the loop filter and phase estimate to the oscillator are not updated, resulting in a vector of angle estimates that is no longer uniformly sampled in time. To adjust the final estimate to be on a uniform time sample grid if desired, we may choose from a variety of techniques including, but not limited to, interpolation, resampling, or functional approximation.

One important connection here betweeen the random sampling CS-PLL (explained in Chapter 8) and other CS-PLLs is that we can treat the use of democracy in a CS-PLL as producing random sampling; in fact, in the random sampling CS-PLL, our implmentation of democracy is trivial, we just adjust the mean sampling frequency of our loop components based on the percentage of samples dropped. We note that we still see the cross-term noise in the phase detector when using the random demodulator or similar systems, rather it is the oscillator and loop filter that have different characteristics. For example our integrator is rescaled proportional to the change in time between each sample relative to the average sampling period. If we are dropping only a few measruements, the effects are small. And even when democracy is applied advantageously as in [28], only roughly 14% of the measurements are lost from saturation, not 50% or more. One disadvantage of this correction is that the majority of the measurements will still continue to occur at a slightly higher sampling rate than the mean. Hence if the parameters are loop filter parameters are particularly sensitive in the operating region, we substantially alter the results. Another downside is that our mean sampling frequency likely will not be an integer factor of the orignal Nyquist frequency in this case, unlike a random-sampling PLL with some sampling schemes such as uniform additive random sampling.

6.4 Interference Signals

Phase-lock loops are designed to lock onto a relatively narrowband signal; noise outside the band of interest must be addressed for the CS-PLL. Because we often have an estimate of the carrier frequency, we can easily apply a traditional bandpass filter around the signal of interest before using a traditional PLL. If we know the location in the spectrum of the signal before acquiring with the compressive sampler, we could apply an analog bandpass filter as in the traditional case and proceed with processing through a CS-PLL. However, the motivations for using CS often preclude filtering beforehand. This may be because we do not know initially where any signal or inteferer is, but find out later. Alternatively, we may wish to monitor the entire spectral band with the same set of data while using the PLL in parallel with other processing. Filtering of the compressive sampled data can be done as shown in [11]. We can decompose the signal x as $x = x_S + x_I$, where x_S is the desired signal and x_I is interference noise. The procedure is different than standard bandpass filtering. Rather than design taps to implement an FIR or IIR filter for a given sampling frequency, a projection matrix P is computed that will map the spectral components of the interfering signal to the nullspace (Py is the filtered signal). In essence we are removing columns of the sampling matrix $\Phi\Psi$ that correspond to the unwanted components.

The interference cancellation matrix is written as

$$P = I - \Phi_J \Phi_J^\dagger, \tag{6.1}$$

where $\Phi_J^\dagger$ denotes the pseudoinverse $\Phi_J^\dagger = (\Phi_J^* \Phi_J)^{-1} \Phi_J^*$. In this instance Φ_J must include the J columns of a basis transform Ψ to the frequency domain, not just the random demodulator' ± 1 structure. Although a Fourier basis is simplest, using J discrete prolate spheriodal sequences modulated to the applicable frequencies is actually better for handling frequencies not quantized to a grid [45]. Although any interference signal does not live in $\mathbb{R}^N$ for a bandlimited length N signal, it lives quite close to the low-dimensional subspace spanned by the first J discrete prolate spheriodal sequences. Each band of interference is cancelled by producing a set of these basis functions and modulating them to the correct frequency with a cosine centered in the band. We can remove multiple interferers by repeatedly applying interference cancellation to bands one at a time, or more simply by concatenating individual basis transforms Ψ_i into one Ψ before computing P.

The projection matrix P may be dense. The density implies that a block-wise algorithm is necessary, making it very difficult to implement interference cancellation in real time. Given the way the CS-PLL was constructed by added a compressive sampler after the oscillator, it would suggest that the cancellation matrix P may need to be added to the loop after the sampler. Because P cannot easily be implemented on a per-sample basis like the random demodulator, this would introduce inordinate

amounts of delay in the loop.

More simply however, P is an orthogonal matrix. Hence

$$\begin{aligned} \langle Py, Pv \rangle &= v^T P^T P y \\ &= w^T P^2 v \qquad (6.2) \\ &= w^T P \qquad (6.3) \\ &= \langle Py, v \rangle \qquad (6.4) \end{aligned}$$

As applying the projection matrix to both input y and reference v is equivalent to operating on only one of the two for a standard inner product (when a moving average filter is present), we make the simplest choice and remove interference cancellation from the loop. (The same fact implies that when performing detection in the radio system with a compressive matched filter, we do not need to adjust the compressive sampling of our reference signals after performing interference cancellation initially.)

We also could design a matrix A to exactly remove x_I and preserve x_S by solving the system of equations

$$A[\Phi_{T_I} \Phi_{T_S}] = [0\ I] \qquad (6.5)$$

where T_I denots the support of x_I and T_S denotes the support of x_S. This is an underconstrained system that could allow applying additional constraints as well. A may or may not be an orthogonal projection depending upon its effects on the components of x not in T_I or T_S.

Finally, we can also show that applying an interference cancellation matrix P to remove interference does not drastically affect the support of our signal. Suppose Φ is a compressive sampler with RIP $O(K_S + K_I)$,where K_S is the cardinality of S ($|S|$) and K_I is $|I|$. If we let z be in the interference support such that z is orthgonal to x and defined as $(I - P)\Phi x_S = \Phi z$, then

$$|\langle (I - P)\Phi x_S, \Phi x_S \rangle| = |\langle \Phi z, \Phi x_S - 0 \rangle| \tag{6.6}$$

$$\leq \delta \|x_S\|_2 \|z\|_2 \tag{6.7}$$

$$\leq \frac{\delta \|\Phi x_S\|_2 \|\Phi z\|_2}{1 - \delta} \tag{6.8}$$

where δ is the RIP isometry constant (and we note that $\langle x, z \rangle = 0$ since they are orthogonal).

Because $(I - P)$ is an orthognal projector and thus $|\langle (I - P)\Phi x_S, \Phi x_S \rangle| = \|(I - P)\Phi x_S\|_2^2$, we can manipulate the expression further and substitute z to obtain

$$\frac{\|(I - P)\Phi x_S\|_2}{\|\Phi x_S\|_2} \leq \frac{\delta}{1 - \delta} \tag{6.9}$$

showing that the projection P can only slightly affect our signal of interest x_S.

6.5 Processing of PLL Output

While reconstruction returns compressively sampled signals to the Nyquist-rate, output of the PLL remains at the low-rate. This must be considered when designing filters or other successive components in a receiver. For example, the frequency response of any de-emphasis filter designed to invert a pre-emphasis filter (applied at the Nyquist rate to the message signal at the transmitter) must be adusted accordingly. Furthermore, with PLLs that output non-uniform samples, we need to take extra caution when filtering, as traditional FIR and IIR filters assuming uniform spacing. One simple way is to apply a non-uniform $\rightarrow$ uniform DFT and then use a traditional inverse DFT to obtain a uniformly spaced signal before filtering.

6.6 Normalization

Some compressive recovery algorithms work regardless of the scaling of the matrix. In the case of the compressive PLL, we will need to normalize the sampler so that the loop gain is not too large resulting in an unstable system. An automatic gain control mechanism is often used in practice to rescale the signal as necessary for real-time environments [22]. However, this generally assumes that changes in amplitude are either slow or happen infrequently, giving the system enough time to compensate. In the case of Bernoulli matrix elements, as in the standard random demodulator, the power per row is fixed and an AGC is sufficient. If we use a Gaussian random demodulator for example however, it may be that the entries are on average normal-

ized, but entries in each row are not normalized. We would really like each row of sampling coefficients p_m (corresponding to compressive sample m) to have roughly equal norm, and hardware calibration is one solution. Another way to handle this is using a dual channel system, for example multiplying the input by two oscillator producing sine and cosine waveforms at the current phase estimate and then using an inverse tangent function to combine the two channels. An example diagram of this system is shown in Figure 6.3.

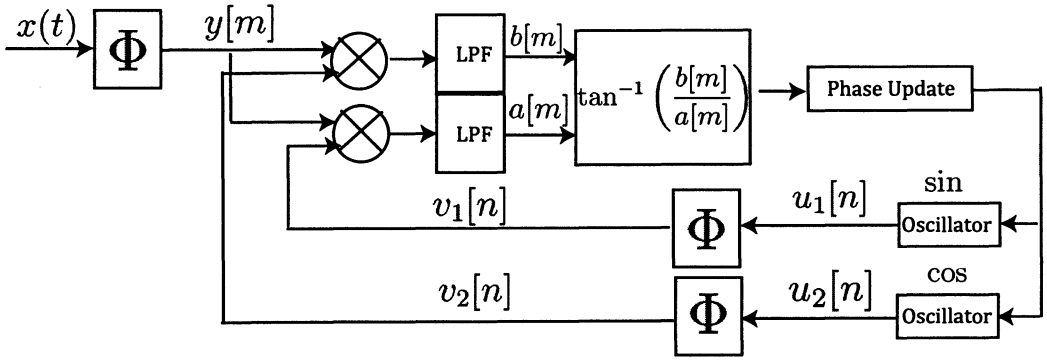


Figure 6.3: CS-PLL designed to handle unnormalized samplers.

Chapter 7

Simulations

We utilize a frequency modulation (FM) application framework to simulate the performance of a compressive PLL. Tracking clock signals or other modulations (such as phase modulation (PM)) requires no modification to the design. We assume that the analog-to-digital converter is ideal except in Chapter 7.3 and use a second-order loop with filter described by Eq. 2.3 for most of the simulations.

7.1 Initial Simulations

Although traditional PLLs generally are not designed with a moving average filter in the loop, the motivation for the CS-PLL was originally based upon this. Hence we demonstrate the PLL works with a moving average filter, originally with length 128 in the Nyquist rate implementation and reduced in length by the compression ratio. A first order IIR filter with response

$$H(z) = \frac{1}{1 - 0.01\frac{N}{M}z^{-1}} \tag{7.1}$$

was then used to update the phase. The system was simulated using a noisy FM modulated audio file originally sampled at 11.625 MHz. Although the results were still decipherable for even a compression ratio of 128, we show results at a more reasonable ratio of 8 in Figure 7.1. We note that initially the CS-PLL struggles to match the input compared to the traditional PLL both when the message signal amplitude is very low and the system has not had time to lock. With the additional noise introduced by the CS-PLL, transients such as these are expected. Another simulation with similar parameters, except that two channels (sine and cosine, combined with arctangent) were used to demonstrate the capability to handle normalization issues, is shown in Figure 7.2.

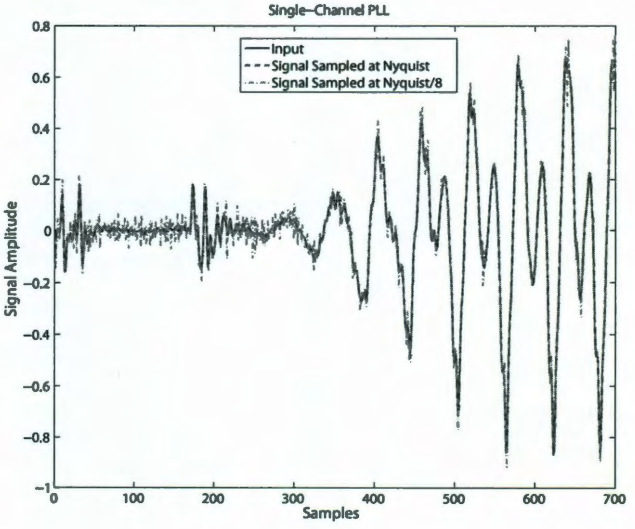


Figure 7.1: Example input and output of an FM modulated audio file demodulated using a traditional and CS-PLL (with compression ratio of 8). The CS-PLL does have more difficulty with transients.

If we treat our message signal as the differentiated phase output (*ie* frequency),

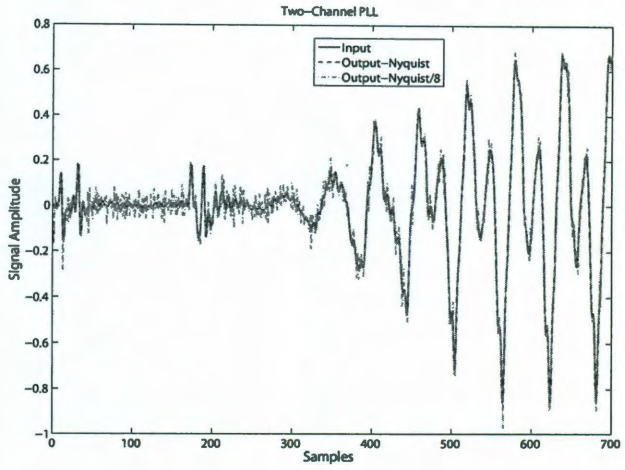


Figure 7.2: Example input and output of an FM modulated audio file demodulated using a traditional and CS-PLL (with compression ratio of 8). The CS-PLL does have more difficulty with transients. Here an additional channel is used in the traditional PLL and CS-PLL combined with arctangent in the phase detector to handle normalization of the sampler.

rather than the phase itself, the effects of compression become far more obvious. In Fig 7.3, we show the output of the traditional PLL and its compressed version, as well result after differentiation. While the phase plots appear roughly the same, the differentiated version appear far different, underscoring the importance of smoothing in a CS-PLL.

We also show that the CS-PLL with a moving average filter can converge to the correct frequency for small enough initial deviations, though it is not as robust as a second order system. Traditional and CS-PLL output are both shown in the upper plot of Fig. 7.4 (with the noisier version the compressed result) for the input in the lower plot. The carrier frequency was set randomly and the oscillator frequency was set as 98% of the carrier frequency. As shown, the output adjusts for this frequency

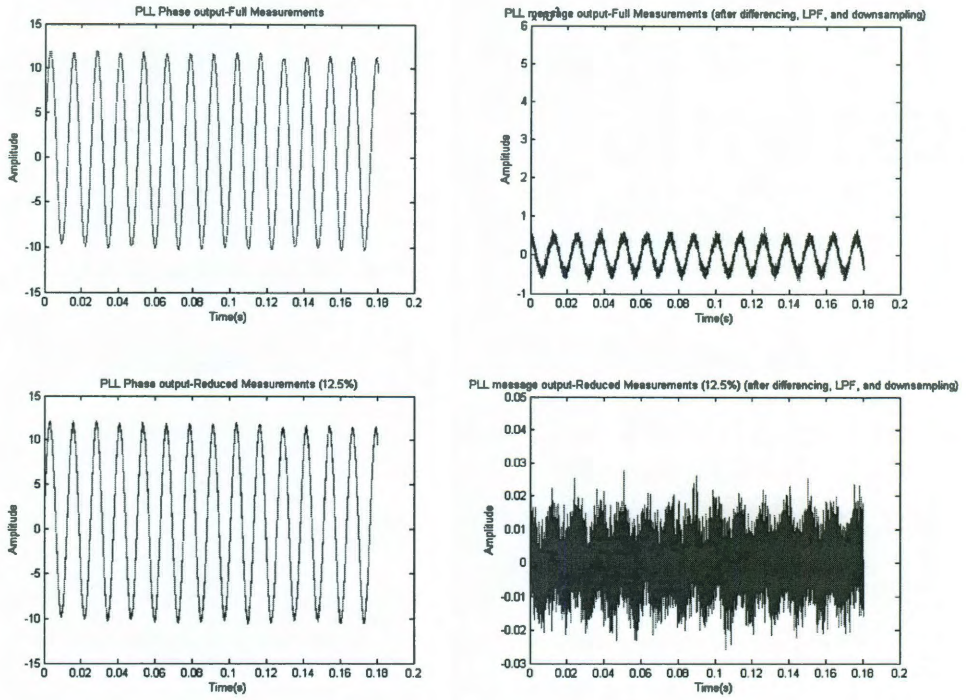


Figure 7.3: Output of PLL operating on audio signal before and after differentiation and filtering.

offset and locks onto the message, tracking the characteristics with a small delay. The only other difference between the input and outputs is an offset in mean value produced as the system converges.

7.2 Second-Order Loop Results

We now test a more standard implementation of the CS-PLL with a second order IIR filter to show that it can maintain lock given the correct carrier frequency. For experiments, we use a sampling rate of 2.048MHz and a oscillator frequency of 120kHz.

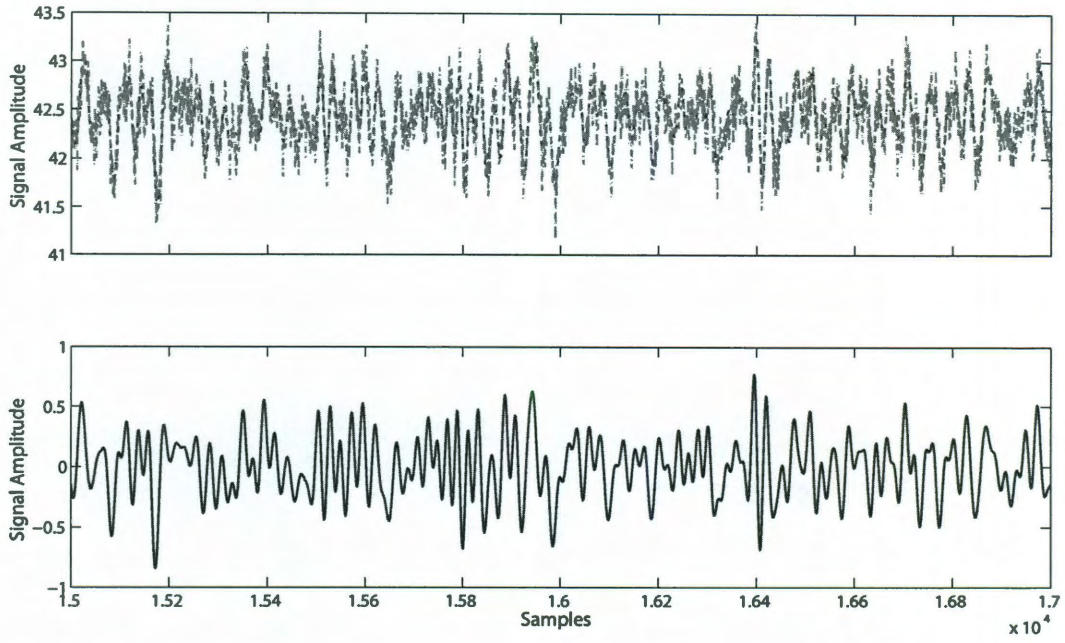


Figure 7.4: The upper plot shows a compressively sampled PLL (dashed line) and traditional PLL (solid line) output with an FIR and a first order IIR filter. The lower plot shows the input, which does not have the delay or offset present in the outputs.

We input an FM modulated 2.5kHz signal with frequency deviation of 1.6kHz compressively sampled into the PLL and restrict the loop filter to a bandwidth of 10kHz. We compare the performance of the CS-PLL to a traditional PLL, so we ignore design factors that are common to both. Pre-emphasis and de-emphasis are not used, and we do not tweak the loop filter to optimize for a particular input. Data is sampled using a random demodulator with ± 1 taps.

Performance is shown using the output SNR of the traditional and CS-PLL for various input SNR and compression factors averaged over 25 trials. Note that the

performance of the traditional PLL corresponds to a logarithmic compression factor of N/M of 0, shown at the left edge of the plots. We do not use an MMSE error criterion due to the inherent delay in discrete-time PLLs. Output SNR is measured by dividing the signal power by the noise power in the frequency spectrum. In Figure 7.5, we compute the noise power over the full spectrum. In the case of the Figure 7.6 and all other performance plots to follow, we compute the noise power over an average 250 Hz band around the signal of interest, basing this on the assumption that other noise can be removed. For low noise on either the message signal or after FM modulation, performance is almost solely dependent on the compression ratio. Here we show a plot of performance varying the input SNR of the FM modulated signal.

We see an initial sharp drop in the SNR when compression is used due to the introduced noise, but then it smooths to a more gradual 3 dB per factor of 2 compression due to aliasing. Fortunately, the initial drop introduced by compressive sensing becomes much less severe as the input SNR declines and we consider more practical scenerios. Also, understandably, the characteristics of the SNR curves when considering noise over the full spectrum or over the narrow signal band are very similar, with the primary difference just being a constant factor. In fact, this supports the notion that Gaussian noise remains over the full spectrum at the output of the CS-PLL.

As mentioned earlier in our analysis, we also test the CS-PLL when an additional offset factor is added to the phase detector based on our current phase estimate derived

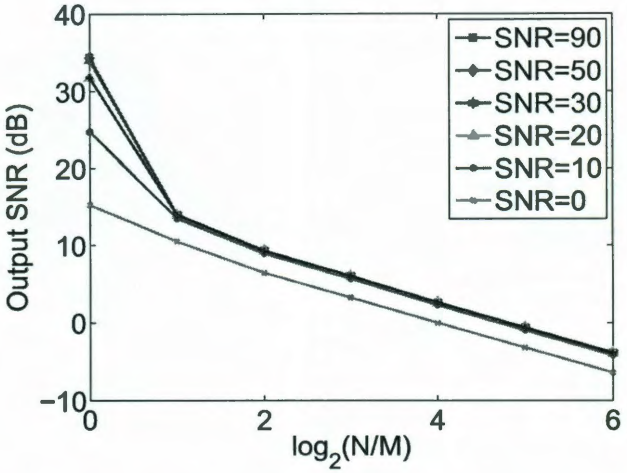


Figure 7.5: Output SNR (over full spectrum) when varying the carrier SNR of the FM signal)

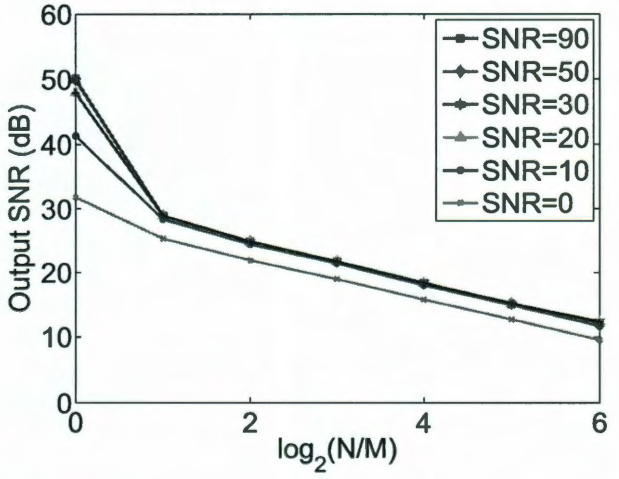


Figure 7.6: Output SNR (over message signal bandwidth) when varying the carrier SNR of the FM signal)

with our maximum likelihood estimator in Eq. 4.6. Fig 7.7(a) shows the results if we add this addtional factor to our system, whereas Fig 7.7(b) includes no additional factor. As we can see the offset minimally increases output SNR at high input SNR. In the more realistic low input SNR regime, this difference is reduced, supporting

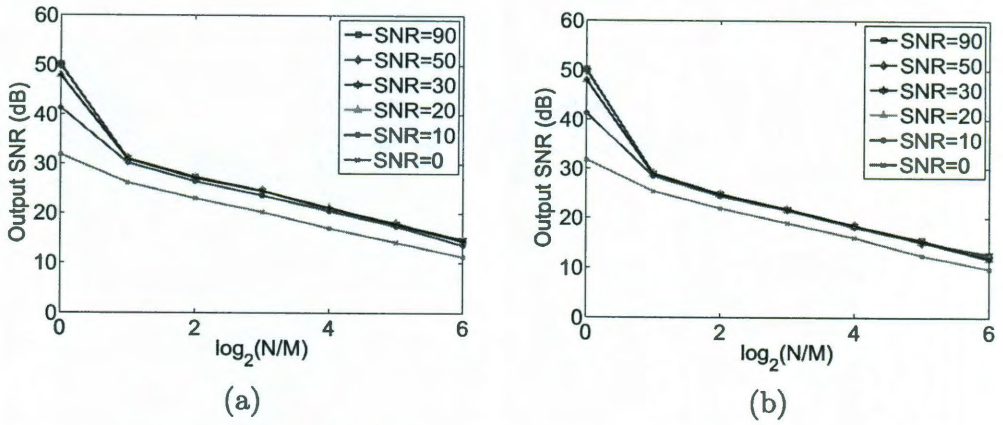


Figure 7.7: Output SNR (over message bandwidth for (a) bias added to the phase detector (b) no bias added to the phase detector

the notion to ignore the offset term. Since the traditional PLL does not have any cross-terms and terms at twice the carrier frequency are filtered out by the low-pass filter in the loop, no offset is added in that case. Presumably, we could attempt to add an additional term at twice the carrier frequency, but this is justifiably not done in practice.

Besides maintaining lock, the other key performance characteristic of a PLL is its ability to acquire lock given a slightly incorrect starting frequency. Therefore, the next test is to assume that the CS-PLL starts with a close but slightly incorrect frequency and adjusts to the proper frequency. This is possible because we are using a 2nd order system. For experiments, we use the same setup as before, except with an initial oscillator frequency of 120kHz while the signal's carrier frequency varies by 5 or 10% in either direction. We also adjust the gain in the loop in one case as well. Incorrectly initializing the oscillator frequency with an error of 5 or 10% from the input carrier frequency only minimally affects the performance as shown in Fig. 7.8

for high loop gains, but can negatively impact as the loop gain is lowered for increased compression as shown in Fig. 7.9. This is largely due to transients as the system will take more time to adapt.

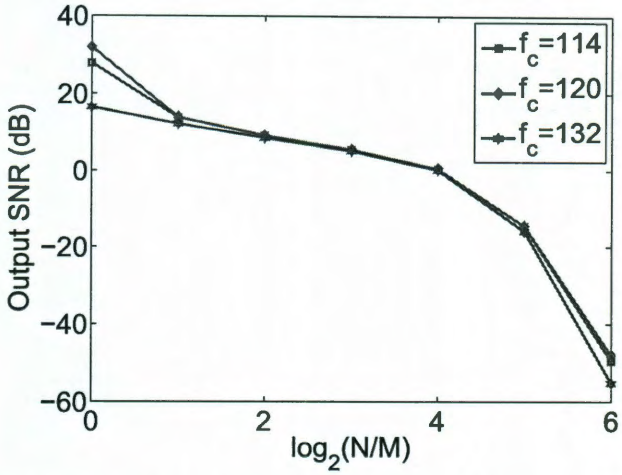


Figure 7.8: Varying the initial estimate of the carrier frequency results in small performance differences for higher loop gains, however we see significant degradation in performace as we take too few measurements (compression grows to a factor of 64).

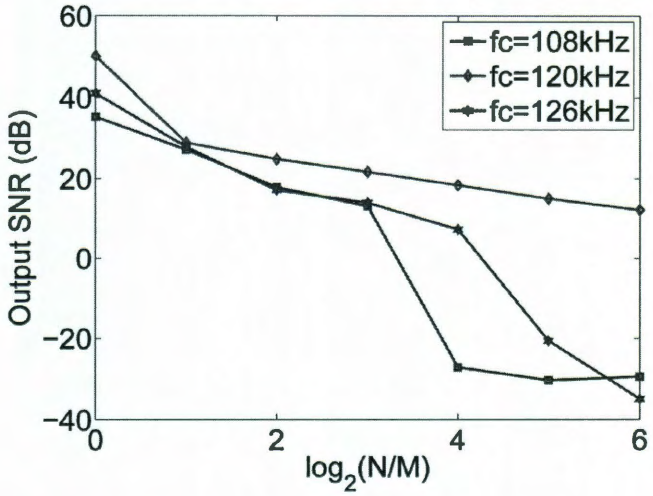


Figure 7.9: Varying the initial estimate of the carrier frequency results in larger performance differences for lower loop gains, as it is particularly prone to transient effects.

Although very frequently when using the random demodulator system, ± 1 are

used as the chipping sequence, we could also construct a sampler with Gaussian coefficients using a random seed. Similar to the previous simulations, SNR results were computed for a 2.5 kHz frequency modulated signal. We notice a slightly better performance using normalized Gaussian taps, (as the crossterm noise is reduced), but slightly weaker performance using unnormalized Gaussian taps (because this essentially varies the gain of the PLL at each time instant). Results are shown in Figures 7.10 and 7.11.

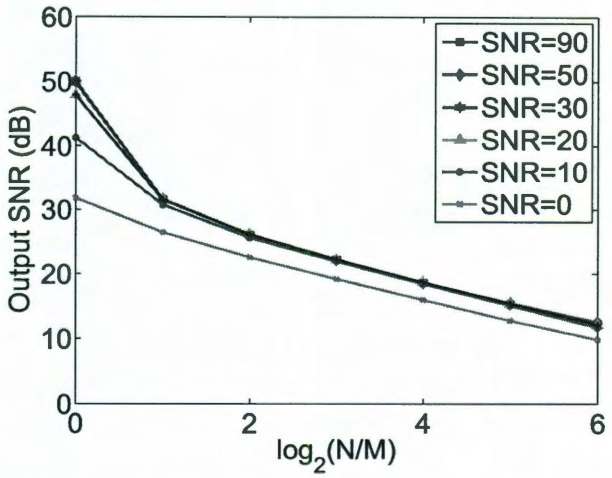


Figure 7.10: Output SNR (over message signal bandwidth) when varying the carrier SNR of the FM signal using a random demodulator with normalized Gaussian taps)

Although in many cases it may be difficult to use a complex CS-PLL without an efficient known Hilbert transfrom in the compressive domain, the performance benefits are quite significant for high input SNR if we can. Results are shown in Figure 7.12 for a similar scenerio as before except using a complex PLL directly extracting the angle in the phase detector. For hight input SNR, the output SNR is much higher than the real case, and in fact we find a small amount of compression is actually good

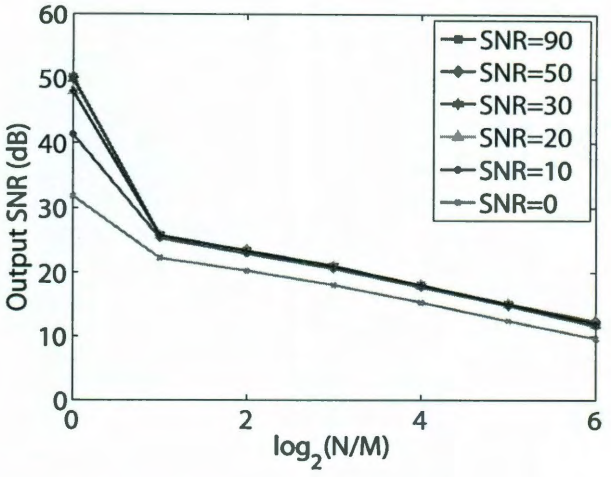


Figure 7.11: Output SNR (over message signal bandwidth) when varying the carrier SNR of the FM signal using a random demodulator with unnormalized Gaussian taps

as it acts as a smoothing filter. As noise increases, the advantage of working in the complex domain becomes smaller however.

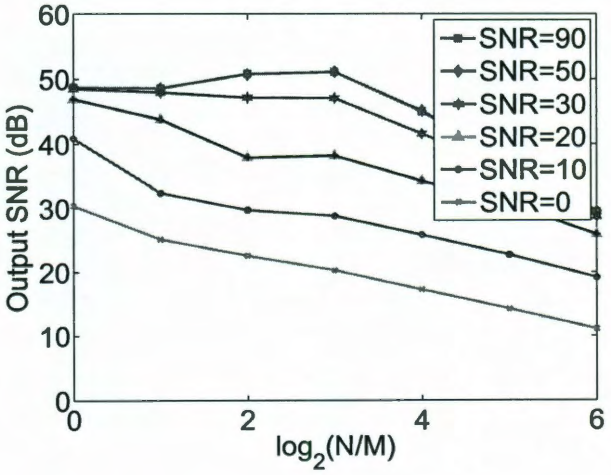


Figure 7.12: Output SNR of a complex CS-PLL (over message signal bandwidth) when varying the carrier SNR of the FM signal using a random demodulator with)

7.3 Applications

We can also consider the results of a system using more practical input signals. The spectrum where a voice signal might appear is simulated in one example, and in another we use digital MSK signals to compute a bit error rate (BER). Each of these is simulated using a 9 bit random demodulation system with pseudorandom sequence of length 40000, more than enough to ensure highly unncorrelated output given that our compression factors are on the order of 100 or less. In addition, a simulated TV interference signal that is 40dB stronger than our signal of interest is removed through intereference cancellation, showing negligible performance degradation to the system after removal. We continue to use a second-order loop in these simulations.

It is important to ensure that the CS-PLL does not smear energy in the message signal across the band of interest, for example with a voice signal. One SNR metric that has been used in the past within some communities is the noise-power ratio (NPR) [18]. Neither the details of the metric nor our adaptation are particularly important, rather we use this as a way to demonstrate an important characteristic that the CS-PLL retains. The NPR test uses a signal with a deep notch in a flat spectrum bandpass noise signal. The original application was in telecommunication or other multiplexing systems to determine the nonlinear effects of multiple channels with a narrow separation region. Instead we consider a narrow notch in a noise message signal which is then frequency modulated.

We find that the notch is very well-preserved at the output of the CS-PLL, on-par

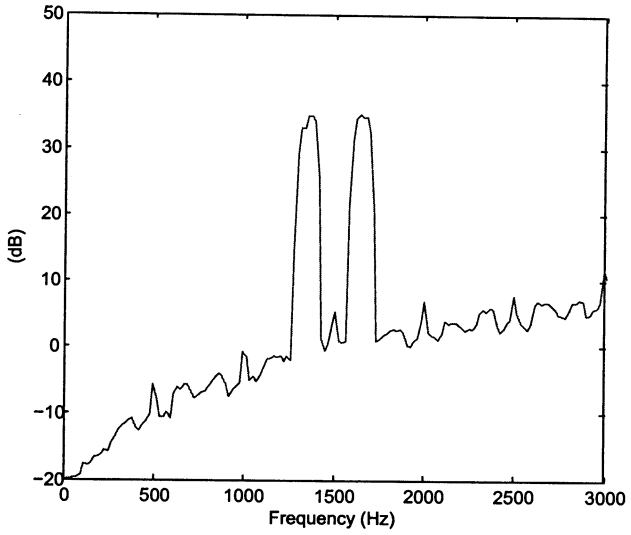


Figure 7.13: Notches in the message signal are well-preserved by the CS-PLL

with the noise floor produced by the CS-PLL as discussed throughout this paper. An example is shown in figure 7.13 where we have signal compressed by a factor of 8. With a 200 Hz notch centered within two signal bands of 100 Hz each at 1500 Hz, we find that the energy within the notch remains 32.5 dB below either signal band.

We can compute a bit error rate(BER) of our system for 500000 bit trials for varying input SNR as measured by the effective number of bits. Effective number of bits is calculated as SNR with an additional term of 10 log(BW/Data Rate). Results are shown for a compression rate of 8 in Figure 7.14, and we can see that the BER falls rapidly with an increase in the effective number of bits. For example, we see that a bandwidth $BW = 3500$, data rate $DR = 2400$, and $SNR = 14.59$, yields an effective number of bits $\frac{E_b}{N_0} = SNR + 10\log(BW/DR) = 16.23$ and results in a BER of 8.77×10^{-5}.

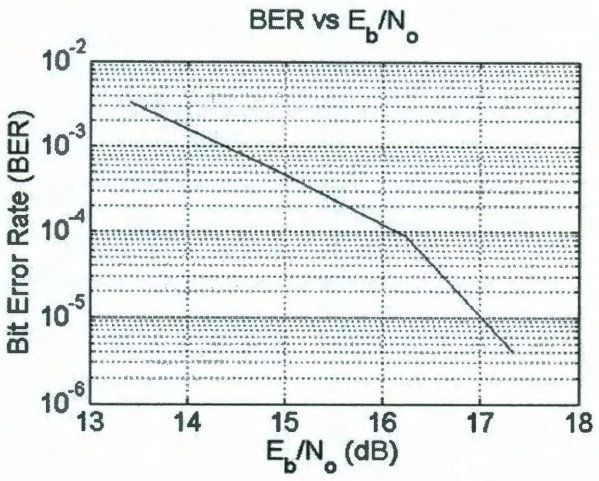


Figure 7.14: Plot of BER vs ENOB for a CS-PLL

7.4 Output

Although the previous results were generated with compression rates of powers of 2, we are not constrained to that. We apply the CS-PLL to an input signal of 25 dB input SNR with a compression ratio of 20. The input signal used was a simulated high power cordless phone to demonstrate that the system works with far more general signals. As Figure 7.15 indicates, the output of the CS-PLL closely resembles the result of FM demodulation using a Hilbert transform and phase unwrapping, even for a case of 25 dB input SNR.

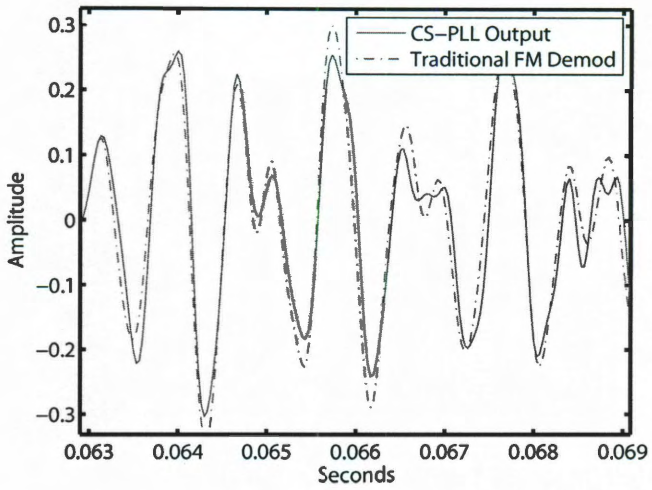


Figure 7.15: CS-PLL output compared to traditional FM demodulation of the Nyquist-rate samples.

Chapter 8

Random Sampling PLL

We can adapt the CS-PLL for the random sampler as introduced in [37]. The random sampler is designed to handle aliasing by spacing the samples (pseudo)-randomly. When we apply a random sampler to the input and oscillator in a PLL, we are simply modifying the time at which the input and oscillator send samples into the phase detector.

The authors of [37] use an additive random sampling scheme through much of their numerical analysis that sufficiently reduces aliasing but limits the deviation from uniform sampling. In their work, the input signal is sampled at instants $t_{i+1} = t_i + T_i$, where T_i is a random period with discrete uniform distribution defined as $T_i = (M + r_i)\delta$. δ corresponds to the original Nyquist period time step, M is the minimum period, and r_i is a discrete uniform random variable on the interval $[0, R]$ such that the average sampling frequency is $f_M = ((M + R/2)\delta)^{-1}$. Constraining R reduces the variance in instantaneous frequency from the mean sample frequency. We instead just choose random samples based on a given mean frequency without constraining this variance, and find that at the compression levels we are working

with that the results are sufficient. In theory it could be possible to have a very large time gap in sampling followed by many small ones. For example, sampling patterns of $100, 1, 1, 1, 1, 1, 1, 1, 1, 2\mu s$ and $11, 11, 11, 10, 9, 12, 11, 13, 8, 14\mu s$ both have mean periods of $11\mu s$, but the latter is much closer to unform and would work better with uniform filtering. This large time gap would essentially imply that information is lost in that gap, and then once the second sample arrived it would perceive a much lower sampling rate than in actuality. However, we should be able to recover quickly in either case, especially if we scale our integrator coefficient appropriately.

Adding the random sampler to the feedback loop has two major tradeoffs versus the random demodulator:

1. No cross-term noise is introduced.

2. Processing of non-uniformly spaced samples is more difficult and computationally expensive.

Downsampling still occurs in the loop. However a uniform Nyquist sampling model is no longer appropriate since we are not extracting information from all Nyquist-rate samples.

We can make simple adjustments to the loop components as follows:

- Phase Detector: This is the same as a traditional PLL, we continue to multiply two sinusoidal terms together

- Loop filter: Create a time-varying filter

- Oscillator: Update the phase and output sample at randomly spaced times.

Further expounding on the loop filter, we note that a traditional loop filter in the PLL assumes uniform sampling. The taps are weighted with the assumption that uniform samples will be provided. With the random sampler however, we have non-uniform samples entering the feedback loop, implying that this assumption is no longer true. To be able to filter the samples accurately we could use a time varying filter and interpolate the taps at each sample. Because the spacing is constantly changing this interpolation would need to be repeated for each sample entering the loop. While this may be doable for an FIR filter, it is very computationally expensive. Furthermore constantly interpolating a loop filter is prone to introducing numerical instabilities. Rather we would like to continue to use a more common IIR model. In fact, the simple solution is to scale the integrator gain by the fraction of time from the previous sample relative to the mean sampling rate. Hence for large gaps of time we assume a bigger change, whereas for very small gaps of time we minimize the change in our integrator. A similar scaling adjustment by the time gap is performed on the output of the loop filter. An additional $2\pi\frac{f_o}{f_s}\Delta n$ term is added, where Δn_m reprsents the number of samples at the Nyquist rate acquired from sample random sample $m-1$ to m. The accumulated phase for sample m can thus be written as $2\pi\frac{f_o}{f_s}n_m + 2\pi\sum_{k=0}^{m}(C_2\theta[n_k] + \frac{C_1}{C_2}\sum_{l=0}^{m}\theta[n_l]\Delta n_l)\Delta n_k$, where Δn_k denotes the Nyquist sample number n of the low rate rate sample k.

What we do gain here is that there is no additional noise added to the system

due to cross-terms as in the case of the random demodulator; each sample out of the phase detector is simply the product of two sinusoids. Because we are still largely oversampling the narrowband signal enough, we can in theory obtain an almost noise-free signal, though the non-linearity of the PLL phase detector still applies.

We note here that the random sampler PLL does not suffer from aliasing at all. Unlike the random demodulator PLL, where signals that would alias to baseband were removed by the sharp nulls of the accumulate and dump filter model, we find that the sine terms are modulated with an additional phase of $2\pi\Delta fr$, where Δf is the discrete-time frequency difference between our signal of interest and aliasing signal, and r is a random integer. Instead we find that the strong aliasing signal has been spread as it enters loop filter.

However, one of the disadvantages of the random sampling PLL is that any further processing of the PLL also continues to be on random samples. Whereas the discrete time fourier transform has been optimized for speed, a non-uniform DFT is quite slow. If we directly apply a uniform FFT to the result, we see quite a bit of spectral smearing, leading to substantially lower SNR calculations than desired. One possible solution is to perform sinc interpolation on the data to obtain uniformly spaced samples for further processing. However, a more efficient solution is to perform a non-uniform DFT in blocks, and then apply the inverse (uniform) FFT on the result. This also then gives us the opportunity to follow with further filtering using standard FIR or IIR filters. An iterative non-uniform DFT method for more accuracy was developed

by Liepins [29], but given the sample rates that these receivers are operating, this would not work for practical applications. Rather we simply contruct a DFT matrix with entries $e^{-j2\pi t_k f_n}$ for a $K \times N$ matrix, where t_k denotes the times of the K samples and f_n denotes the N uniformly spaced frequencies.

What we notice is that the dependence on the compression factor disappears in the spacing of the spectrum and in SNR performance results. We perform some of the same simulations as in the random demodulator case using a message signal at 2.5 kHz with the same loop bandwidth. In Fig 8.1(a), we show the effects of varying the input SNR of the carrier signal. Although we obtain a drop in performance from the initial compression, we see that the SNR stays fairly constant after that. Fig 8.1(b) confirms that the performance is very similar if we use a uniform filter in the loop instead of updating our filter coefficients at each iteration, and Fig 8.1(c) shows the performance decline for initial frequency offsets. As in the case of the random demodulator, the drop in performance can be explained by the transient effects. Once the loop locks, we obtain performance comparable to starting at the correct frequency. For higher levels of compression however, we cannot achieve lock quickly enough for the number of input samples given.

Since we are observing a single Nyquist-rate sample per measurement, extending the traditional Kalman filter to the random-sampling PLL is quite simple. We only need a single phase state (and the one frequency state) as in traditional PLL case. The only thing that changes is our observation matrix H_m, with it simply indicating

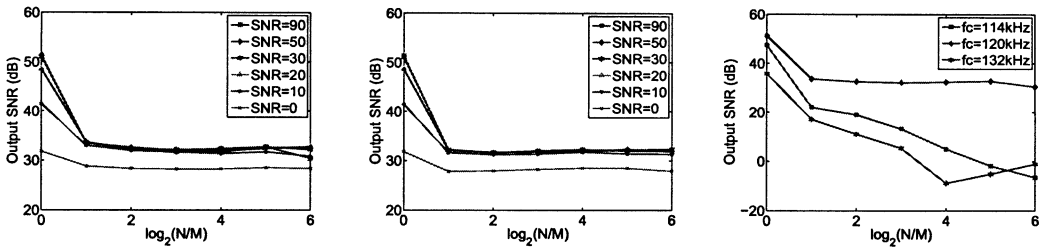


Figure 8.1: Output SNR of a Random Sampling PLL (RS-PLL) for different input SNRs for (a) time-varying varying loop filter coefficients based on sampling times (b) constant loop filter coefficients (ie uniform loop filter) (c) varying initial carrier frequency

whether the phase was observed at sample m or not. Thus at times when a random sample is acquired

$$H_m = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

whereas otherwise

$$H_m = \begin{bmatrix} 0 & 0 \end{bmatrix}.$$

When no sample is acquired, the Kalman filter gain for that iteration is identically zero. This implementation is essentially a degenerate version of random demodulator case; we use the coefficients of the compressive sampler at each iteration, but because we have only one non-zero coefficient per compressive sample, we need only a single phase state. Unlike the Kalman filter for the random demodulator with multiple phase states, assuming unccorrelated states is much more reasonable in this case.

Chapter 9

Additional CS PLLs

So far we have shown results for a few major compressive sampler implementations, namely the random demodulator and random sampler. Here we provide an introduction to several alternative CS-PLL implementations.

9.1 CMUX

For the CMUX compressive sampler, we find that the CS-PLL implementation is trivial. In the case of the CMUX architecture, the compression factor comes from the combination of multiple bands, rather than using a linear combination of samples within a band. The compressive sampling matrix is simply the concatenation of diagonal matrices with diagonal elements alternating between ± 1. Once we have filtered any narrowband inteference (including any other signals of interest) in the combined band using interference cancellation, the perceived sampler for our signal of interest becomes just a modulation by ± 1. Our sampler model in the loop must simply reverse the effects of this modulation by multiplying by the same ± 1 (if α is a Bernoulli random variable, then $\alpha^2 = 1$). An example implementation is shown in Fig. 9.1

where we are demodulating an FM signal in channel j. Since spectra in other bands are modulated with an independent ± 1, the application of this modulation continues to spread their energy (if α and β are independent Bernoulli random variables with equal probabilities, their product $\alpha\beta$ is also a Bernoulli random variable with equal probabilities). Hence we can treat these overlapping spectra as additional AWGN noise on our carrier signal. Unlike the use of random demodulators there are no additional cross-terms, and unlike the random sampler there are no issues with deviating from uniform sampling. However it is still quite advantageous to apply interference cancellation to remove other signals in the band as the noise can be overwhelming just as in the traditional PLL case. A Kalman filter can be constructed simply by using

$$H_m = \begin{bmatrix} p_m^{(j)}[m] & 0 \end{bmatrix}$$

in place of

$$H_m = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

in the Kalman filter implementation for a traditional PLL. As in the random sampling Kalman filter case, we do not need additional phase states.

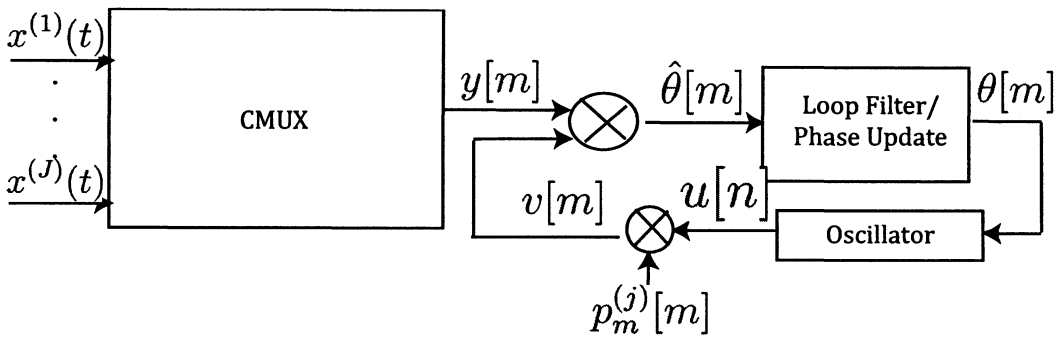


Figure 9.1: Example CS-PLL implementation for the CMUX compressive sampler. Here we are demodulating channel j.

9.2 Other Variations

9.2.1 Maximum Correlation PLL

One alternative CS-PLL that was attempted consisted of maximizing the correlation over time between the input and and sinusoidal signals by comparing a compressively sampled input against several compressively sampled reference sinusoids at different frequencies. This differs from a traditional PLL that tries to minimize the correlation over time between the input signal and an orthogonal reference signal. Each channel still used an ideal multiplier and each result was filtered with the same (FIR) low-pass filter design. At each iteration, the maximum correlation was chosen. In this case, we cannot just adjust the phase proportional to the filtered correlation value, as it no longer has the same meaning. Rather, if the input signal was more highly correlated with a lower frequency signal, the frequency of the reference signal was decreased by some constant parameter ϵ for sample $m+1$ (ie $f_{m+1} = f_m - \epsilon$), whereas if the input signal was more highly correlated with a higher frequency signal,

the frequency of the reference signal was increased (ie $f_{m+1} = f_m + \epsilon$). To ensure lock, a third state was added in which the PLL was held at the same frequency if the input signal was most highly correlated with a reference signal at the current frequency estimate (ie $f_{m+1} = f_m$). A similar thresholding effect could be added to other versions of the CS-PLL by ignoring any small phase changes out of the loop filter. However it is particularly important in this case where the frequency estimate is being changed discretely and not directly proportional to the error. The tradeoff in using a thresholding effect is that we cannot track small changes in frequency and realize a signficant lag when the PLL finally determines a frequency change has occurred. An example of this design is shown in Fig. 9.2, where the phase update block choses the maximum correlation response and outputs $\theta = f_m + \epsilon$ to the oscillator.

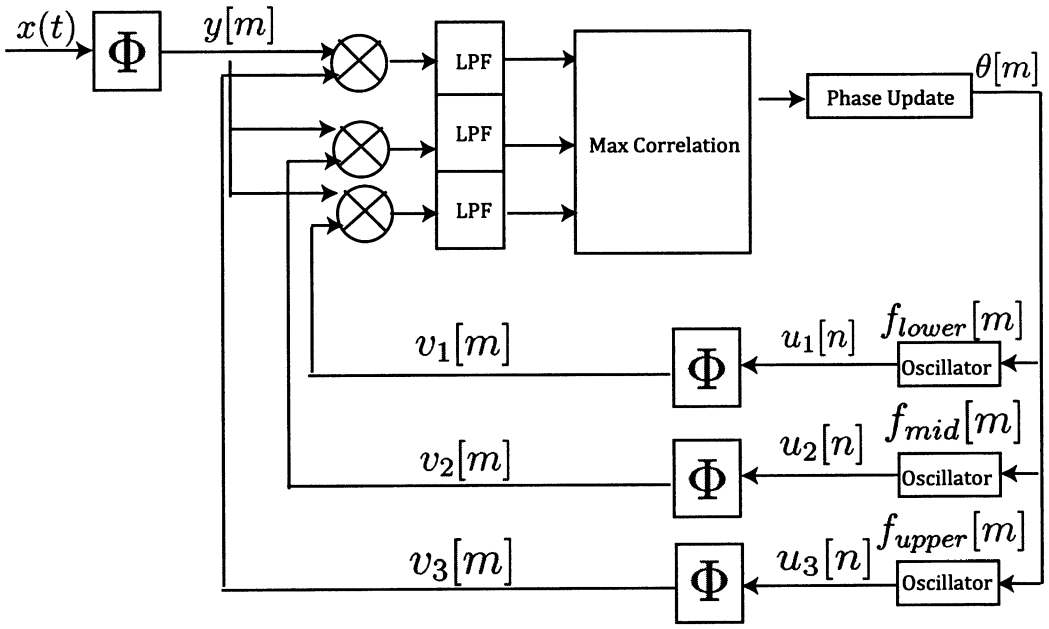
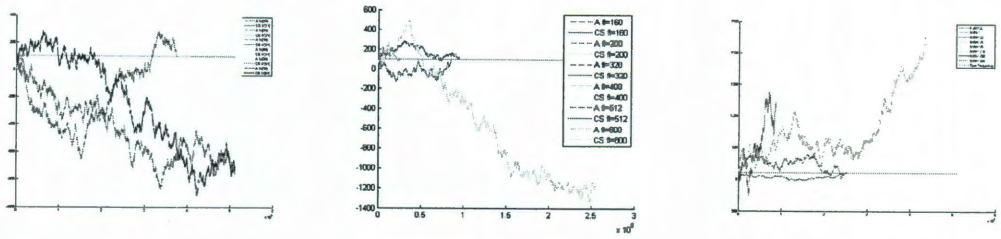


Figure 9.2: CS-PLL relying on maximizing correlation between input and several reference inputs.



(a) (b) (c)

Figure 9.3: Frequency output of PLL maximizing correlation with compression (denoted as CS) vs no compression (denoted as A). (a) Varying ϵ ($N = 1024$), filter length $fl = 320$, compression factor $M/N = 1/8$ (b) Varying filter length (fl), $\epsilon = 2\pi/N$, compreesion factor $M/N = 1/8$ (c) Varying compression factor $N/M = 1/8$, filter length $fl = 320$, $\epsilon = 2\pi/N$.

However what we find is that while this consistently converges for a noise-free input signal with no compression it is far less reliable for any level of compression. There were no consistent patterns in convergence as we adjusted the step parameter ϵ, the length of the filter, or even the compression level as soon as any compression was added. If our initial frequency estimate was very close to the true value, this did dramatically increase our convergence rate. On the other hand, as soon as we started some distance away, the noise-free input signal would converge whereas many of the compressed signals would not in the given timespan. Here the delay introduced by the filters is even more detrimental than in the traditional PLL framework.

Several example results are shown here, though they are not meant to be extensive. As we can see in Fig. 9.3, the traditional PLL converges so fast to the true frequency of 101 for all parameters that it is not even obvious in the figure.

9.2.2 BPSK

If we know that our input is a low sample rate BPSK, we could use the output of the CS-PLL to distinguish the bits after additional processing. Unfortunately, the CS-PLL is not designed for low oversampling in severe case like PSK or FSK where the frequency/phase changes are quite sudden and large. However, we can still see a noticeable difference in the output for two different bits with the CS-PLL. Simple processing such as squaring the output or using an absolute value would allow us to distinguish between the two cases. In Fig. 9.4 we show the nominal phase output of first order traditional and CS-PLLs. In the traditional case, constant phase indicates a lock on a bit (0), whereas, ramp change indicates a frequency difference indicating a different bit (1). For the CS-PLL case, we notice the lock as before, but the nominal value does not demonstrate the ramp change. We also notice that the system has far more difficulty transistioning from bit to bit.

9.2.3 Pseudo-random sequence generation

Although quantization was already addressed in the scope of the PLL systems components, there also is an issue of the randomness of the sampling coefficients $p[k]$. If we use pseudorandom ± 1 sequences the randomness for practical purposes will be limited by the bits in a bit-shift register. Hence some tests were performed initially to verify that the output SNR was not impacted by number of bits producing the sequence. An example plot is shown in Fig. 9.5. We note that loop design had not

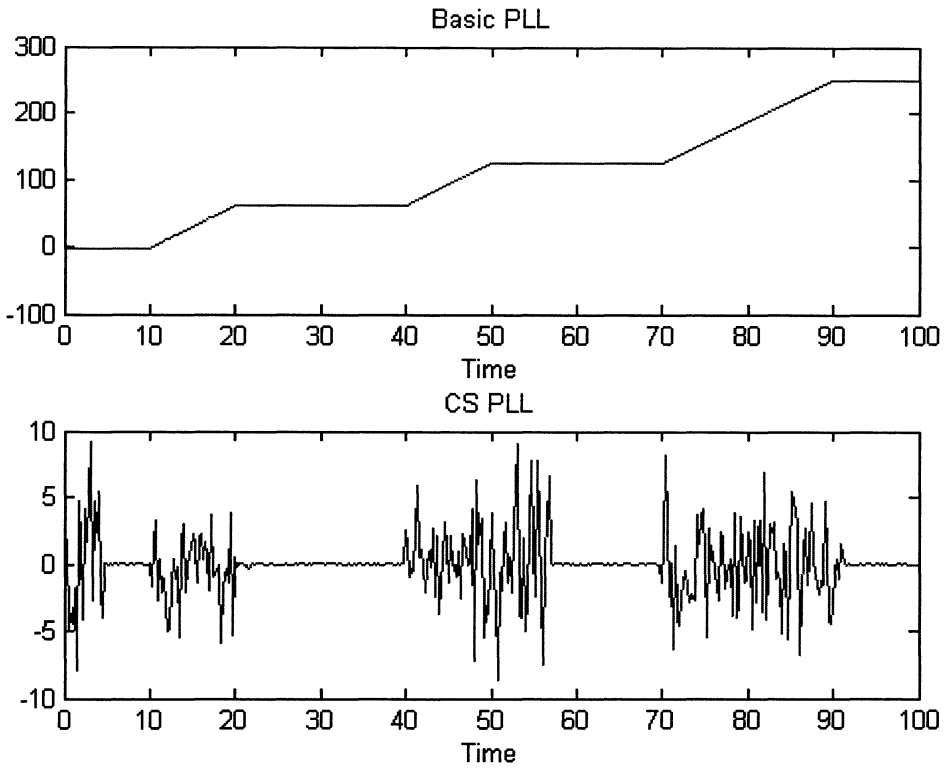


Figure 9.4: Phase Output of a CS-PLL with BPSK. Phase here is the nomina valuel including frequency changes in a first-order system. A traditional PLL will lock to one bit (0) and show constant ramp change for the other (1). On the other hand the CS-PLL does not produce the ramp change, though there is noticeable difference between the two bits.

been optimized at this point, we were still using a first-order IIR phase update with an FIR loop filter. We notice that after enough bits were present the number of bits of randomness does not affect the results. Rather we simply see the 3dB per factor of 2 compression present denoted by the drop in curves as we decrease the percentage of measurements taken (curves shown in the figure are distinguished by the ratio of measurements out of 1024, with 1024/1024 indicating the results of a traditioanal PLL. (The traditional PLL's output SNR was high enough that it is not visible on

the plot.) Surprisingly enough, the plot shows an increase in performance for very few bits, though this can be explained by the oddities that occur as we have so little randomness for small levels of compression. As we increase compression, we do not get nearly as substantially different behavior. Additionally at small levels of randomness the bias towards a single ± 1 tap is more noticeable (since the pure 0 state is often not used in pseudo-random bit-shift registers). Other standard CS algorithms rely on the assumption that reasonable levels of randomness are present anyways, so generally we will not be operating in this region.

9.2.4 Others

In addition to the CS-PLLs that have been mentioned here so far, we note that other variations are possible. For example, multiple loops could be used to track phase and frequency with a frequency tracking loop updating a phase tracking loop as shown in Fig. 9.6. The CS-PLL version would take compressive measurements of each oscillator. Other CS-PLLs include the use of different modulations from the oscillator such as square and triangular wave functions.

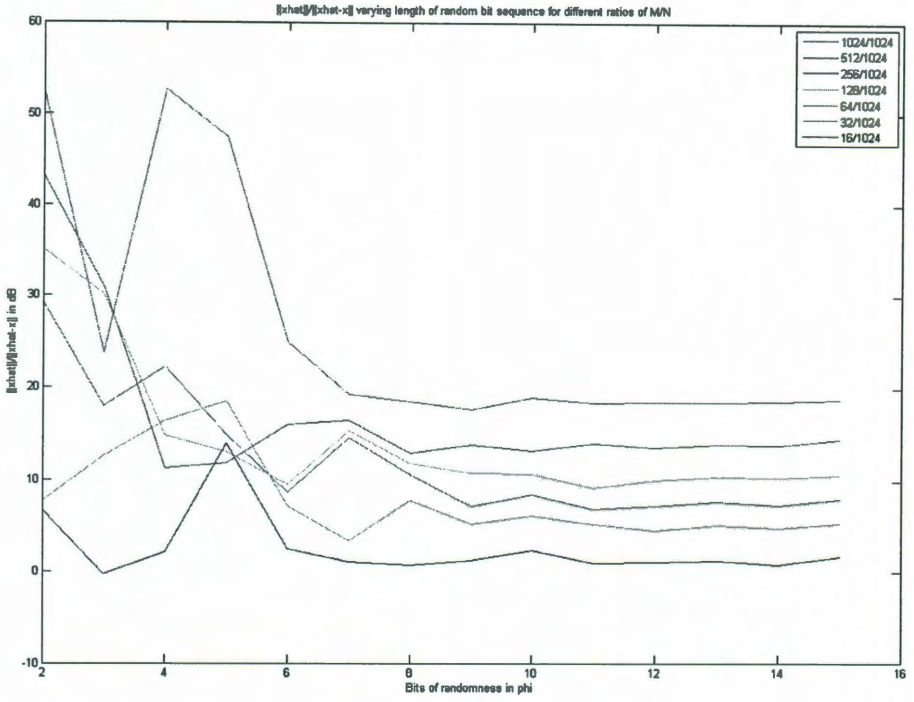


Figure 9.5: Performance for an example first order PLL with varying bits of randomness in the pseudo-random sampling sequence. The region with very low number of bits is an artifact of the bias towards a single bit and also more obvious with smaller levels of compression. Otherwise we observe an expected 3 dB drop as we compress by additional factors of two (compression levels are computed as a fraction of 1024 here).

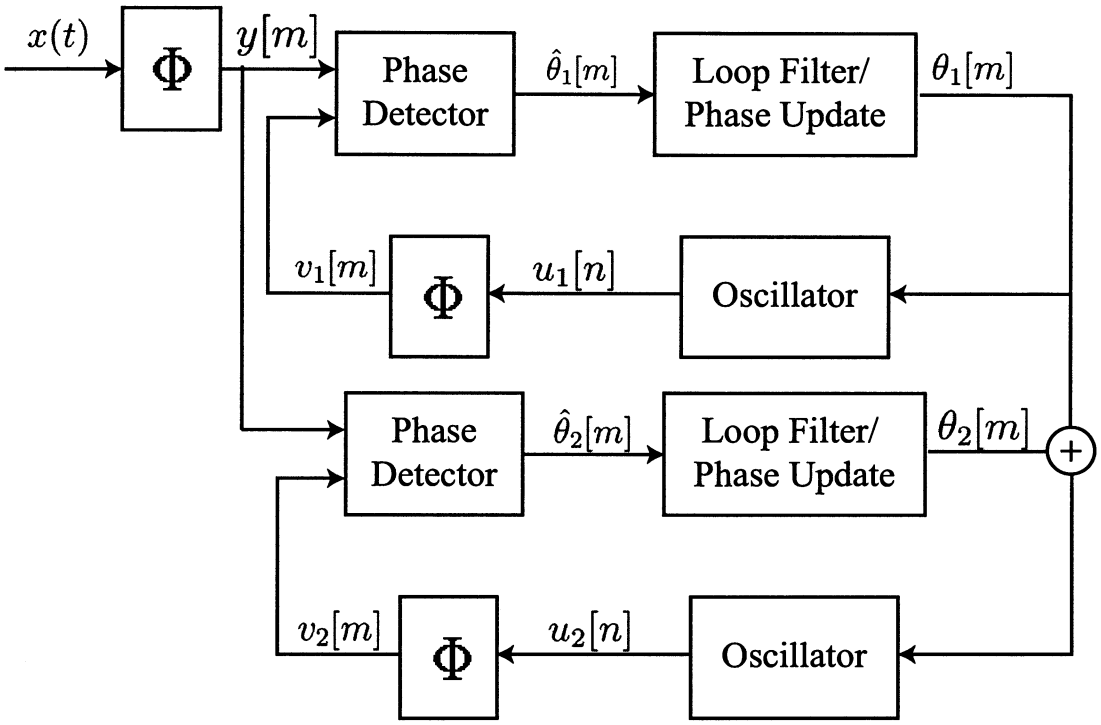


Figure 9.6: Example Dual CS-PLL that uses two loops together, one to update phase and the other to handle frequency changes).

Chapter 10

Discussion and Conclusions

Motivated by CS's approximate preservation of standard inner products, this paper lays the framework for constructing a phase-locked loop for compressive signal reconstruction. The ideas could be extended relatively simply to variants of the PLL that use a multiplier phase detector; applying CS to those variations of phase-locked loops that operate with simple switches or counters without any signal multiplication is much more difficult.

While the PLL was designed with an FM application in mind, the CS-PLL can be used in a wide variety of circumstances. Clock synchronization is another common application for phase-locked loops. Because of noise introduced by the compressive PLL, it is not suited for extremely high fidelity synchronization applications. However, the CS-PLL can be utilized if some noise is acceptable. The random sampling pattern may also help avoid effects of unwanted harmonics that would otherwise alias onto the signal.

Frequency modulation systems are not the sole users of PLLs. Communication schemes using phase modulation and amplitude modulation can benefit from using

PLLs as well. In addition, GPS satellites needing to reduce power consumption in their A/D converters can utilize the CS-PLL. Under some circumstances, even industrial applications such as induction flow meters could use the CS-PLL.

Bibliography

[1] R. Baraniuk. Compressive sensing. *IEEE Signal Processing Mag.*, 24(4):118–120, 124, 2007.

[2] R. G. Baraniuk, M. A. Davenport, R. DeVore, and M. B. Wakin. A simple proof of the Restricted Isometry Property for random matrices. *Const. Approx.*, 28(3):253–263, Dec. 2008.

[3] P. Boufounos and M.S. Asif. Compressive sampling for streaming signals with sparse frequency content. In *Information Sciences and Systems (CISS), 2010 44th Annual Conference on*, pages 1 –6, 2010.

[4] E. Candès. Compressive sampling. In *Proc. Int. Congress of Math.*, Madrid, Spain, Aug. 2006.

[5] E. Candès. Compressive sampling. In *Proc. Int. Congress of Mathematics*, pages 1433–1452, Madrid, Spain, Aug. 2006.

[6] E. Candès and T. Tao. Decoding by linear programming. *IEEE Trans. Inform. Theory*, 51(12):4203–4215, 2005.

[7] E. J. Candès, J. Romberg, and T. Tao. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Trans. Info. Theory*, 52(2):489–509, Feb. 2006.

[8] Volkan Cevher, Aswin Sankaranarayanan, Marco F. Duarte, Dikpal Reddy, Richard G. Baraniuk, and Rama Chellappa. Compressive sensing for background subtraction. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 155–168, Marseille/France, Oct. 2008.

[9] James Crawford. *Advanced Phase-Lock Techniques.* Artech House, 2008.

[10] M. Davenport, P. Boufounos, M. Wakin, and R. Baraniuk. Signal processing with compressive measurements. *IEEE J. Select. Top. Signal Processing*, 4(2):445–460, 2010.

[11] Mark A. Davenport, Petros T. Boufounos, and Richard G. Baraniuk. Compressive domain interference cancellation. In *Workshop on Signal Processing with Adaptive Sparse Structured Representations (SPARS)*, Saint-Malo, France, Apr. 2009.

[12] Mark A. Davenport, Stephen R. Schnelle, J. P. Slavinsky, Richard G. Baraniuk, Michael B. Wakin, and Petros T. Boufounos. A wideband compressive radio receiver. In *Military Communications Conference (MILCOM)*, San Jose, California, Oct. 2010.

[13] D. Donoho. Compressed sensing. *IEEE Trans. Inform. Theory*, 52(4):1289–1306, 2006.

[14] D. L. Donoho. Compressed sensing. *IEEE Trans. Info. Theory*, 52(4):1289–1306, September 2006.

[15] M. Duarte, M. Davenport, D. Takhar, J. Laska, T. Sun, K. Kelly, and R. Baraniuk. Single-pixel imaging via compressive sampling. *IEEE Signal Processing Mag.*, 25(2):83–91, 2008.

[16] M. F. Duarte, M. A. Davenport, M. B. Wakin, J. N. Laska, D. Takhar, K. F. Kelly, and R. G. Baraniuk. Multiscale random projections for compressive classification. In

IEEE International Conference on Image Processing (ICIP), pages VI–161–164, San Antonio, TX, Sept. 2007.

[17] Floyd M. Gardner. *Phaselock Techniques, Third Edition.* Wiley, 2005.

[18] Khaled M. Gharaibeh. On the relationship between the noise-to-power ratio (npr) and the effective in-band distortion of wcdma signals. *AEU - International Journal of Electronics and Communications*, 64(3):273 – 279, 2010.

[19] A. C. Gilbert, S. Muthukrishnan, and M. J. Strauss. Improved time bounds for near-optimal sparse Fourier representations. In *Proc. Wavelets XI at SPIE Optics and Photonics*, San Diego, CA, Aug. 2005.

[20] S. Goldman. *Phase-Locked Loop Engineering Handbook for Integrated Circuits.* Artech House, 2007.

[21] S Hahn. *Hilbert Transforms in Signal Processing.* Artech House, 1996.

[22] C. R. Johnson Jr and William A. Sethares. *Telecommunication Breakdown.* Prentice Hall, 2004.

[23] Sithamparanathan Kandeepan and Sam Reisenfeld. Phase detector models and their performances for if/baseband frequency recovery for complex envelope based dsp implemented pll. *IEEE ICCS*, 2002.

[24] S. Kirolos, J. Laska, M. Wakin, M. Duarte, D. Baron, T. Ragheb, Y. Massoud, and R. Baraniuk. Analog-to-information conversion via random demodulation. In *In Proc. IEEE Dallas Circuits and Systems Workshop (DCAS)*, 2006.

[25] J. Laska, P. Boufounos, M. Davenport, and R. Baraniuk. Democracy in action: Quantization, saturation, and compressive sensing. Preprint, 2009.

[26] J. Laska, S. Kirolos, Y. Massoud, R. Baraniuk, A. Gilbert, M. Iwen, and M. Strauss. Random sampling for analog-to-information conversion of wideband signals. In *Proc. IEEE Dallas Circuits and Systems Workshop (DCAS)*, Dallas, TX, Oct. 2006.

[27] J. N. Laska, S. Kirolos, M. F. Duarte, T. Ragheb, R. G. Baraniuk, and Y. Massoud. Theory and implementation of an analog-to-information conversion using random demodulation. In *Proc. IEEE Int. Symposium on Circuits and Systems (ISCAS)*, New Orleans, LA, May 2007. To appear.

[28] Jason N. Laska, Petros Boufounos, Mark A. Davenport, and Richard G. Baraniuk. Democracy in action: Quantization, saturation, and compressive sensing. *Preprint*, 2009.

[29] V. Liepin'sh. A spectral estimation method of nonuniformly sampled band-limited signals. In *Automatic control and computer sciences*, volume 28, pages 66–73, 1994.

[30] P.T. Boufounos Ashok Veeraraghavan M. Asif, Dikpal Reddy. Streaming compressive sensing for high-speed periodic videos. In *Proc of IEEE International Conference on Image Processing (ICIP)*, pages 3373–3376, Sept 2010.

[31] M. O.Sonnaillon, R. Urteaga, F. J. Bonetto, and M. Ordonez. Random sampling in high-frequency digital lock-in amplifiers. *Reunion de Trabajo en Procsamiento de la Informacion y Control*, 752, Sep 2005.

[32] S.H. Raghavan and J.K. Holmes. Performance of costas and phase locked loops with signal blanking. In *Aerospace Conference, 2005 IEEE*, pages 1524 –1531, 2005.

[33] J. P. Slavinsky, Jason N. Laska, Mark A. Davenport, and Richard G. Baraniuk. The compressive multiplexer for multi-channel compressive sensing. In *Preprint*, Oct. 2010.

[34] M. O. Sonnaillon and F. J. Bonetto. Fpga implementation of a phase locked loop based on random sampling. *Programmable Logic, 2007. SPL '07. 2007 3rd Southern Conference on*, Feb 2007.

[35] M. O. Sonnaillon, R. Urteaga, and F. J. Bonetto. High-frequency digital lock-in amplifier using random sampling. *Instrumentation and Measurement, IEEE Transactions on*, 57(3):616–621, Mar 2008.

[36] M. O. Sonnaillon, R. Urteaga, F. J. Bonetto, and M. Ordonez. Implementation of a high-frequency digital lock-in amplifier. *Electrical and Computer Engineering, 2005, Canadian Conference on*, pages 1229–1232, May 2005.

[37] M.O. Sonnaillon, R. Urteaga, and F.J. Bonetto. Software pll based on random sampling. *Instrumentation and Measurement, IEEE Transactions on*, 59(10):2621 –2629, 2010.

[38] Christian Stimming. Frequency offset tracking in mcma systems. *Technical report, Berkley Wireless Research Center*, May 2001.

[39] Haiyun Tang. Notes on dpll. Technical report, University of California-Berkeley.

[40] J. Treichler, M. Davenport, and R. Baraniuk. Application of compressive sensing to the design of wideband signal acquisition receivers. In *U.S./Australia Joint Work. Defense Apps. of Signal Processing (DASP)*, Lihue, Hawaii, Sept. 2009.

[41] J. Tropp, J. Laska, M. Duarte, J. Romberg, and R. Baraniuk. Beyond Nyquist: Efficient sampling of sparse, bandlimited signals. *IEEE Trans. Inform. Theory*, 56(1):520–544, 2010.

[42] J. Tropp, M. Wakin, M. Duarte, D. Baron, and R. Baraniuk. Random filters for compressive sampling and reconstruction. In *Proc. IEEE Int. Conf. Acoust., Speech, and Signal Processing (ICASSP)*, Toulouse, France, May 2006.

[43] N. Vaswani. Kalman filtered compressed sensing. In *Image Processing, 2008. ICIP 2008. 15th IEEE International Conference on*, pages 893 –896, 2008.

[44] Ashwin A. Wagadarikar, Nikos P. Pitsianis, Xiaobai Sun, and David J. Brady. Video rate spectral imaging using a coded aperture snapshot spectral imager. *Opt. Express*, 17(8):6368–6388, Apr 2009.

[45] M. Wakin and M. Davenport. Discrete prolate spheroidal sequences for compressive acquisition and processing of bandlimited signal. *In Preparation.*

[46] A. Weinberg and B. Liu. Discrete time analyses of nonuniform sampling first- and second-order digital phase lock loops. *Communications, IEEE Transactions on*, 22(2):123–137, Feb 1974.

[47] Lu M. Feng Z. Zhao, S. A phase-locked loop design with external aiding based on kalman filter. In *Proceedings of the 22nd International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS 2009)*, pages 3096–3100, 2009.